SIMULASI ROBOT PATROLI SEDERHANA UNTUK DETEKSI BERBASIS COMPUTER VISION DAN DEEP LEARNING

Disusun Oleh:

N a m a : Ivan Setya Ananda
NIM : 19523232

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
2023

HALAMAN PENGESAHAN DOSEN PEMBIMBING

SIMULASI ROBOT PATROLI SEDERHANA UNTUK DETEKSI BERBASIS COMPUTER VISION DAN DEEP LEARNING

TUGAS AKHIR

Disusun

N a m a : Ivan Setya Ananda

NIM : 19523232

Yogyakarta, 7 Juli

Pembimbing,

(Chandra Kusuma Dewa, S.Kom., M.Cs., Ph.D.)

HALAMAN PENGESAHAN DOSEN PENGUJI

SIMULASI ROBOT PATROLI SEDERHANA UNTUK DETEKSI BERBASIS COMPUTER VISION DAN DEEP LEARNING

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjanadi Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 20 Juli 2023

Tim Penguji

Chandra Kusuma Dewa, S.Kom., M.Cs.,Ph.D. _____

Anggota 1

Rahadian Kurniawan, S.Kom., M.Kom. _____

Anggota 2

Dhomas Hatta Fudholi, S.T., M.Eng.,Ph.D. _____

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Ivan Setya Ananda
NIM : 19523232

Tugas akhir dengan judul:

SIMULASI ROBOT PATROLI SEDERHANA UNTUK DETEKSI BERBASIS COMPUTER VISION DAN DEEP LEARNING

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri dan apabila dikemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya saya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung resiko dan konsekuensi apapun

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 20 Juli 2023

(Ivan Setya Ananda)

HALAMAN PERSEMBAHAN

Assalamualaikum Warahmatullahi Wabarakatuh.

Alhamdulillah puji syukur terpanjatkan kehadirat Allah SWT yang telah memberikan kemudahan, kelancaran, dan juga ridho-NYA sehingga penulis dapat menyelesaikan tugas akhir ini tepat pada waktunya dan selalu diberikan kesehatan dalam proses pengerjaannya. Tugas akhir ini penulis persembahkan kepada berbagai pihak yang berpengaruh dalam proses penulis menjadi orang yang lebih baik, kritis, dan bijak untuk memilih dan menentukkan proses hidup penulis. Pertama penulis persembahkan kepada keluarga tercinta yang selalu memberikan dukungan dan naungan secara fisik, material, maupun emosional sehingga penulis dapat menyelesaikan tugas akhir dengan lancar. Kedua penulis persembahkan kepada jurusan Informatika Universitas Islam Indonesia yang telah memberikan berbagai macam wadah dan dukungan untuk penulis dalam mengembangkan wawasan dan tempat untuk menuntut ilmu sehingga penulis lebih percaya diri untuk menempuh lingkungan yang akan dihadapi selanjutnya. Ketiga penulis persembahkan kepada lembaga khusus Marching Band Universitas Islam Indonesia yang telah mempercayakan penulis untuk mengemban amanah dan tanggungjawab sehingga penulis dapat lebih disiplin dan bertanggungjawab atas tugas-tugas penulis serta telah memberikan naungan dan fasilitas kepada penulis untuk selalu berkarya dalam kondisi dan situasi apapun. Keempat penulis persembahkan kepada sahabat-sahabat penulis sebagai teman cerita yang dapat membuka berbagai sudut pandang penulis dan sebagai teman seperjuangan cerita hidup penulis.

Wassalamualaikum Warahmatullahi Wabarakatuh.

HALAMAN MOTTO

“Jika kamu tidak mengambil risiko, kamu tidak akan dapat menciptakan masa depan!.”
– Monkey D.Luffy (One Piece)

KATA PENGANTAR

Alhamdulillahi rabbil ‘alamin. Segala puji bagi Allah SWT tuhan seluruh alam yang senantiasa selalu melimpahkan rahmat serta rezeki kepada penulis agar terus dapat tumbuh berproses menjadi manusia yang lebih baik dan bertaqwa sebagai hamba-NYA. Shalawat serta salam senantiasa tidak lupa terhaturkan kepada Nabi Muhammad SAW sebagai petunjuk dan pedoman segala sesuatu yang benar dan yang menjauhkan manusia dari jaman kebodohan menuju jaman yang penuh dengan rahmat dengan adanya ilmu pengetahuan yang terbalut dalam agama penuh kasih sayang yaitu Islam sebagai rahmat bagi seluruh alam dengan pedomannya Al-quran dan Al-Hadist.

Atas izin Allah SWT penulis dapat menyelesaikan penelitian dan tugas akhir dengan judul “Implementasi Robot Patroli Sederhana Berbasis Computer Vision dan Deep Learning”, sebagai salah satu syarat kelulusan dalam jenjang S1 Informatika Universitas Islam Indonesia. Dalam proses pengerjaan tugas akhir ini, penulis ingin menyampaikan rasa terimakasih kepada:

1. Orang tua penulis, Bapak Soeminto dan Ibu Dyah Utari sebagai saksi hidup jatuh bangun penulis
2. Kakak kandung penulis, Rifqi Setya Perdana
3. Dosen pembimbing penulis, Bapak Chandra Kusuma Dewa, S.Kom., M.Cs., Ph.D
4. Dosen – dosen pengampu mata kuliah Informatika UII beserta para jajarannya
5. Sahabat-sahabat penulis; Farros, Naufan, Naufal, Zidan, Atul, Kalya
6. Asisten pelatih color guard MBUII; Awal, Aul, Belva, Cahyo
7. Tim teknis MBUII; Mizan, Icak, Wendy, Venny, Daffa, Annisa, Fathur, Shike
8. Anggota color guard MBUII

Yogyakarta, 7 Juli 2023

(Ivan Setya Ananda)

SARI

Kegiatan patroli membutuhkan tingkat konsentrasi dan konsumsi energi yang tinggi, sedangkan makhluk hidup memiliki banyak limitasi dalam melakukan aktivitas yang bersifat reptitif. Makhluk hidup terutama manusia terdapat batasan tingkat konsumsi energi dan persepsi kecerdasan yang berbeda-beda. Robot patroli merupakan salah satu teknologi yang dikembangkan pada aktivitas patroli. Oleh karena itu, penelitian ini bertujuan untuk mensimulasikan robot patroli yang dibangun melalui framework atau perangkat lunak ROS (*Robot Operating System*) yang mampu membangun struktur robot dan lingkungan uji coba simulasi sederhana. Dengan ROS robot akan diuji berbasis computer vision dan deep learning, pada implementasinya robot menggunakan model YOLO (*You Only Look Once*) versi kelima atau YOLOv5. Simulasi akan diujikan terhadap jarak dan skenario. Terdapat tiga skenario yang dilakukan meliputi objek tunggal, objek ganda dan objek berjalan pada model manusia dan mobil. Hasil dari pengujian model manusia diam dan bergerak mendapatkan akurasi di rentang 80% hingga 90% pada jarak 1 – 10 meter dan dalam kondisi terhalang objek lain, sedangkan pada objek model mobil mendapatkan akurasi di rentang 50% hingga 80% pada jarak 1 – 6 meter. Namun, pada objek mobil berwarna tidak netral seperti biru kerap mengalami salah identifikasi objek dan melabeli objek dengan model kelas lain.

Keywords—Robot Patroli, Computer Vision, Deep Learning, YOLO, ROS

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Patroli merupakan "aktivitas untuk berkeliling atau menyusuri area pada interval reguler untuk berbagai macam kepentingan umumnya seperti observasi atau pemeliharaan keamanan"(Portugal & Rocha, 2011). Patroli memiliki berbagai macam target untuk pemantauan dan pengawasan lingkungan, memperoleh informasi, mencari objek dan mendeteksi adanya anomali, yang membutuhkan keterlibatan secara berkala pada setiap poin infrastrukturnya(Portugal & Rocha, 2011). Patroli merupakan hal yang umum dilakukan di seluruh dunia sebagai contoh dalam menekan tingkat keamanan. Prevalensi saat ini Index keamanan di Indonesia sendiri berkisar 46.63% pada Oktober 2022 dan berdasarkan survei data tingkat keamanan di Indonesia meningkat sebanyak 12.60% dari 34.03% pada 3 tahun terakhir yang disebabkan oleh peningkatan pengawasan patroli saat pandemi Covid-19("Asia: Current Crime Index by City," 2022). Namun, menurut survei ("Crime in Indonesia. Safety in Indonesia," 2019) tingkat kekhawatiran kriminalitas di Indonesia terdapat 71.43% terhadap pencurian, penganiayaan dan perusakan properti pribadi.

Kemampuan patroli masih dipengaruhi berbagai macam limitasi: keterbatasan kerja makhluk hidup(Lee, Lin, & Huang, 2011), titik buta pada kamera pengintai(Lee et al., 2011), dan rintangan pada lingkungan(Rostami, Sangaiah, Wang, & Liu, 2019). Salah satu masalah yang paling sering ditemui adalah keterbatasan kerja makhluk hidup. Makhluk hidup memiliki banyak limitasi apabila ditemukan kepada pekerjaan yang melibatkan keterlibatan secara berkala(Hiatt, Harrison, & Trafton, 2011). Contohnya, pada manusia yang memiliki pengaruh emosi dan psikologi dalam menjalankan patroli sehingga dapat menciptakan berbagai tingkah laku yang tidak sesuai(Hiatt et al., 2011). Tidak hanya itu kepercayaan dan pengetahuan yang dimiliki oleh makhluk hidup tidaklah sama(Lopez, Paredes, Quiroz, Trovato, & Cuellar, 2017). Oleh karena itu, perlu adanya pendekatan umum untuk menentukan proses berpikir apa yang dapat membantu pada keterbatasan tindakan manusia sehingga dapat meningkatkan aktivitas berpatroli(Hiatt et al., 2011).

Saat ini berbagai macam pendekatan teknologi telah dikaji dan diterapkan dalam menggantikan peran makhluk hidup dalam bertugas patroli. Teknologi yang telah digunakan hingga saat ini diantaranya adalah kamera pengintai(Lee et al., 2011), drone patrolling(Nguyen, Jenssen, & Roverso, 2018), dan robotic patrolling(Lopez et al., 2017). Robotic patrolling merupakan salah satu pendekatan yang dapat menyamakan proses berpikir manusia dan menyamakan kemampuan ilmu pengetahuan manusia(Basilico, 2022). Robot memiliki kemampuan untuk memantau lingkungan dan mendeteksi(Basilico, 2022).

Dari hasil pemaparan data dan fakta tentang pengaruh teknologi terkait kegiatan berpatroli, maka penelitian ini bertujuan untuk mengembangkan robot patroli sebagai komplementer pada manusia dalam melakukan deteksi. Penelitian ini akan merancang sebuah simulasi dengan robot melalui sebuah perangkat lunak yang dapat digunakan untuk mengevaluasi kinerja patroli robot dan pengetahuan apa yang bisa atau harus dilakukan untuk mengoptimalkan pengembangan robot patroli.

1.2. Rumusan Masalah

Rumusan masalah pada penelitian ini yaitu:

a) Bagaimana membangun sistem yang dapat membantu mendeteksi objek lingkungan sebagai bagian dari aktivitas patroli
b) Apakah computer vision dan deep learning pada robot patroli dapat menghasilkan akurasi yang baik pada deteksi objek lingkungan

1.3. Batasan Masalah

Batasan masalah pada penelitian ini yaitu:

a) Aktivitas patroli yang dimaksud hanya dibatasi pada deteksi dan identifikasi objek lingkungan
b) Aktivitas patroli hanya dilakukan dalam lingkungan simulasi komputer
c) Algoritma computer vision yang digunakana adalah YOLO versi 5

1.4. Tujuan Penelitian

Tujuan penelitian ini yaitu:

a) Mensimulasikan robot sebagai alat deteksi pada kegiatan patroli untuk membantu manusia

b) Melakukan pengujian terhadap robot patroli terkait kemampuan mendeteksi lingkungan

1.5. Manfaat Penelitian

Manfaat penelitian ini yaitu:

a) Mengetahui apakah robot patroli yang dikembangkan mampu menjadi komplementer kegiatan manusia dalam berpatroli untuk mendeteksi lingkungan
b) Mempermudah manusia dalam melakukan kegiatan patroli yang dilakukan secara berkala
c) Menguji kemampuan robot patroli dalam mendeteksi dalam berpatroli

1.6. Metodologi Penelitian

Perancangan pada penelitian ini dibagi menjadi tiga tahapan.

PRA-PELAKSANAAN

Tahapan ini berisikan identifikasi masalah dan analisis metodologi, pada identifikasi masalah penulis mengangkat topik patroli sebagai masalah yang bisa dikembangkan dalam penelitian dengan metode penggunaan robot deep learning dan computer vision yang difokuskan kepada deteksi objek terhadap manusia dan objek pribadi.

PELAKSANAAN

Tahapan pelaksanaan dibagi menjadi dua tahap. Pada tahap pertama yaitu pengumpulan data. Pada pengumpulan data terdiri dari menyiapkan lingkungan penelitian yang melaksanakan pembentukan ROS sehingga dapat berperilaku seperti makhluk hidup dan pengumpulan data deteksi objek dari dataset COCO. Kemudian tahapan pengujian model robot terhadap model deep learning yang sudah disiapkan.

PASCA PELAKSANAAN

Setelah model diteliti maka tahapan terakhir adalah mengevaluasi dan menganalisis hasil yang telah diujikan, kemudian menyimpulkan apakah robot dapat menjawab rumusan masalah yang telah ditentukan dalam melakukan patroli sesuai dengan hasil data penelitian.

1.7. Sistematika Penulisan

Sistematika penulisan dan penyusunan laporan tugas akhir ini terdiri dari beberapa bab, yang mencakup gambaran dari kesuluruhan masalah dan penyelesaiannya. Berikut sistematika penulisan yang terbagi dalam 5 bab:

BAB I PENDAHULUAN

Pada bab ini berisi pembahasan latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian dan sistematika penulisan.

BAB II LANDASAN TEORI

Pada bab ini berisi pembahasan mengenai tinjauan terhadap penelitian yang sudah pernah dilakukan berhubungan dengan apa yang dirancang dan diimplementasikan serta teori dasar yang digunakan berhubungan dengan sistem dalam mengimplementasikan robot patroli sebagai alternatif patroli dari keterbatasan kerja pada makhluk hidup.

BAB III METODOLOGI PENELITIAN

Pada bab ini berisi uraian tentang perancangan dan pembelajaran robot terkait kemampuannya dalam melakukan aktivitas patroli.

BAB IV HASIL DAN PEMBAHASAN

Pada bab ini akan dibahas tentang hasil model dan pembelajaran robot patroli terkait metodologi yang digunakan sebagai implementasi dan analisis pada simulasi robot patroli.

BAB V KESIMPULAN DAN SARAN

Pada bab ini merupakan bab terakhir yang akan membahas kesimpulan dan saran terhadap penelitian yang telah dilakukan pada tugas akhir

BAB II
LANDASAN TEORI

2.1. Implementasi Robot Patroli

Ambisi dalam pengembangan robot patroli adalah untuk memungkinkan robot mengambil keputusan cerdas berdasarkan masukan dari computer vision dan dapat menyusun strategi gerakan yang tepat serta dapat meniru tingkah laku manusia pada saat patroli setelah akrab dengan kondisi lingkungannya(Maram, Vishnoi, & Pandey, 2019). Seiring berjalannya dengan urbanisasi dan digitalisasi yang cepat, maka terjadilah pertumbuhan eksponensial skala besar terhadap populasi manusia ke lokasi-lokasi umum seperti hotel, pasar, mal, tempat bermain dan tempat transportasi umum. Terdapat kemajuan besar di bidang robotika dan computer vision yang terintegrasikan melalui perangkat lunak dapat memberikan sebuah solusi alternatif untuk melawan kriminalitas yang sering terjadi di lokasi-lokasi umum atau sektoral publik(Maram et al., 2019).

ROS (*Robot Operating System*) sebuah perangkat lunak yang dapat menambahkan elemen-elemen seperti path planning dan environment perception yang apabila di kombinasikan dengan kecerdasan neural networks dapat meniru tingkah laku manusia dalam melakukan aktivitas salah satunya ketika sedang melakukan patroli(Maram et al., 2019).

2.1.1. ROS(*Robot Operating Syste*m)

ROS (*Robot Operating System*) adalah framework middleware populer yang digunakan dalam pengembangan perangkat lunak robotika(Santos, Cunha, & Macedo, 2019). ROS bukanlah sistem operasi, tetapi menyediakan layanan yang diperlukan dan dirancang untuk sekelompok server yang heterogen untuk mengendalikan perangkat low-end, menyebarkan fungsi, mentransmisikan pesan dan mengelola perangkat low-end. ROS dikenal dengan desain terdistribusi dan modular, serta menyediakan arsitektur berlapis yang fleksibel(Kang, Lee, & Shin, 2020).

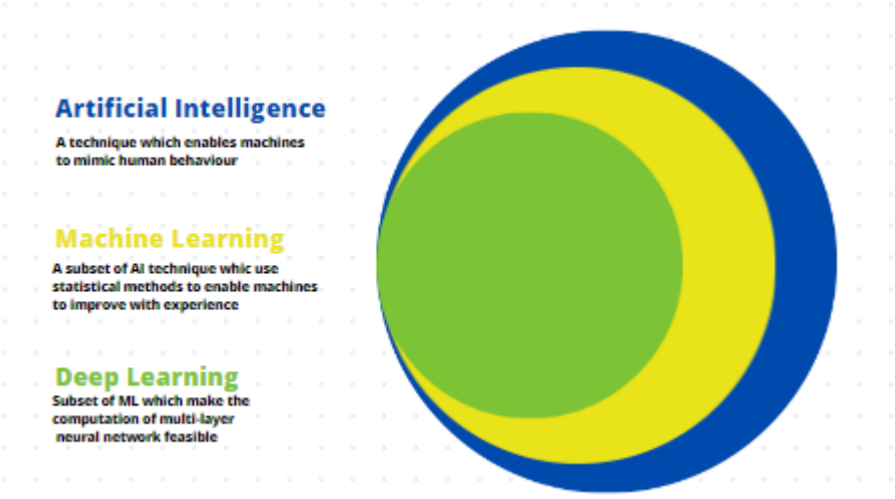
ROS adalah sekumpulan proses berbasis. ROS yang direpresentasikan sebagai grafik arsitektur dapat menerima dan menggabungkan data sensor, mengontrol keadaan, penjadwalan, aktuator dan pesan lainnya. ROS merupakan framework yang menyediakan berbagai macam tools

dan libraries untuk menulis dan menjalankan software robot, ROS memiliki berbagai macam fitur untuk membantu pengembang dalam melakukan seperti penyampaian pesan, komputasi terdistribusi, penggunaan kembali kode dan penerapan algoritma canggih untuk aplikasi robotik(Lentin Joseph & Jonathan Cacace , 2018).

Penggunaan ROS telah membuat terobosan baru untuk komunitas pengembang dalam menyediakan alat, infrastruktur dan praktik untuk membuat aplikasi dan robot dengan upaya penyediaan libraries yang mudah digunakan untuk berbagai kemampuan seperti navigasi, manipulasi, kontrol, dll. Penggunaan ROS dalam dunia robotika memiliki dampak dan manfaat yang besar di kalangan pengembang. Kesulitan terbesar dalam robotika adalah membangun komunikasi yang lancar antara berbagai komponen. Namun, ROS dapat menjembatani celah tersebut dengan cara function yang diperlakukan sebagai node dan setiap node dapat mempublikasikan ke dalam topik topik pilihan yang diteruskan sebagai pesan(Maram et al., 2019). Manfaat lainnya pada saat sebelum kita berinvestasi untuk membeli perangkat keras robotika yang mahal, ROS menjadikan hal ini sebagai manfaat untuk mensimulasikan terlebih dahulu sebelum terjun membangun perangkat keras.

2.2. Deep Learning dan Computer Vision

Deep Learning memiliki beberapa definisi di dalam literatur, bisa didefinisikan sebagai "*inference of model parameters for decision making in process mimicking the understanding process in the human brain*" atau singkatnya "*brain-like model identification*"(Hassaballah Mahmoud & Awad Ismail, 2020). Deep learning merupakan inferensi data dalam machine learning dan termasuk ke dalam operasi utama penggunaan artificial intelligence modern penjelasan terkait perbedaan artificial intelligence, machine learning dan deep learning dapat dilihat pada Gambar 2.1.



Gambar 2.1. AI vs ML vs DL

Source: ((Suzuki, Machado, Matsuzaka, & Yashiro, 2023)

Optimasi pada deep learning selalu berkembang dibandingkan dengan menggunakan tradisional machine learning. Keuntungan menggunakan deep learning dapat mengekstraksi fitur fitur kompleks pada gambar, audio, dan teks. Data yang sudah terkumpul pun bisa selalu diupdate secara real time menggunakan algoritma back propagation. Banyak arsitektur berbeda yang cocok digunakan untuk menyelesaikan masalah kompleks yang beragam(Voulodimos, Doulamis & Protopapadakis, 2018).

Salah satu metode dalam deep learning terdapat computer vision. Computer vision adalah bidang ilmu komputer yang bertujuan agar komputer dapat melihat dan memahami dunia melalui penglihatan seperti manusia(Suzuki et al., 2023). Computer vision bisa secara langsung mengekstraksi fitur pada citra asli. Sehingga, dapat mengatasi subjektivitas dan keterbatasan ekstraksi fitur pada metode tradisional. Penggunaan end-to-end structure dapat menyederhanakan proses pengenalan dan melakukan pemecahan masalah pada kasus patroli yang terjadi karena dapat mengekstraksi atribut alami objek secara jelas(Voulodimos et al., 2018).

Tugas utama dari computer vision meliputi klasifikasi citra, lokalisasi objek, deteksi objek, dan segmentasi(Alam et al., 2022). CNN(*Convolutional Neural Networks*), Deep Boltzmann Machines, Deep Belief Networks dan Stacked Denoising Autoencoder adalah beberapa skema

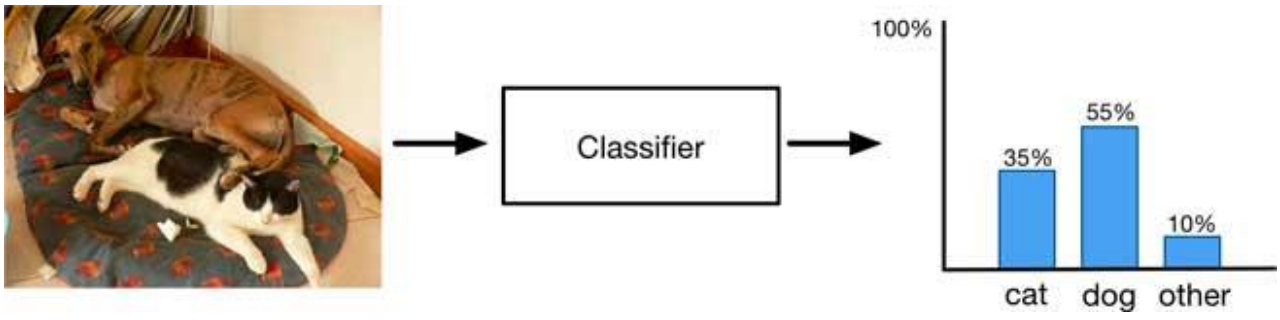
pembelajaran mendalam yang paling signifikan digunakan dalam masalah visi komputer. Selain itu, deep learning telah diusulkan dan didemonstrasikan untuk berbagai aplikasi yang berkaitan dengan citra dan analisis pola difraksi(Alam et al., 2022).

Berdasarkan penerapan computer vision dan deep learning tidak hanya dapat menghemat waktu dan tenaga, tetapi juga dapat menganalisis kemungkinan bahaya secara cepat dan menjadi penerus komunikasi yang cepat apabila terjadi ancaman di sekitar daerah patroli.

2.2.1. Object Detection

Object detection adalah teknik dari computer vision dalam menemukan objek yang menarik dalam sebuah gambar(Hollemans. M, 2018). Computer vision memiliki peran penting dalam pengembangan sistem pakar, tujuan object detection secara otomatis menentukan keberadaan dan lokasi spasial tertentu objek dalam gambar. Sehingga, dapat secara efektif membantu sistem otomasi dan kecerdasan seperti robot untuk memahami dunia dengan lebih baik dengan menganalisis secara otomatis sinyal digital dari kamera dan memberikan informasi penting untuk pengambilan keputusan pada navigasi, langkah gerak, kontrol, dll(Cao, Liao, Song, Chen, & Li, 2021).

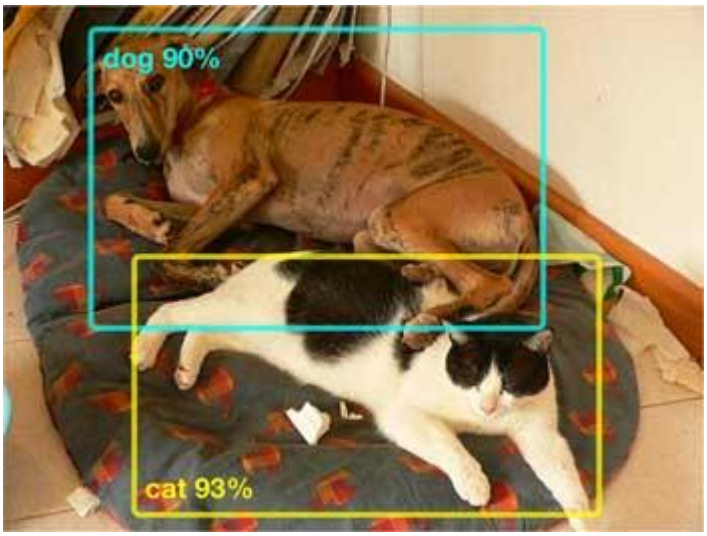
Object detection lebih kompleks daripada klasifikasi. Klasifikasi hanya memberitahu apa "subjek utama" dari gambar tersebut, sedangkan pada deteksi objek dapat menemukan beberapa item untuk mengklasifikasikannya dan menempatkannya di dalam gambar. Satu masalah yang akan dihadapi adalah ketika gambar pelatihan berisi mulai dari nol hingga lusinan objek dan model menampilkan lebih dari satu prediksi(Hollemans. M, 2018).



Gambar 2.2.Metode Klasifikasi

Source: (Hollemans. M, 2018)

Pada Gambar 2.2. pengklasifikasian mengenali bahwa gambar tersebut mengandung objek seperti kucing dan anjing dalam jumlah tertentu, tetapi hanya batasan tersebut yang bisa dilakukan dari klasifikasi. Sedangkan pada Gambar 2.3 Sebuah model deteksi objek akan memberi tahu dimana masing-masing objek berada dengan memprediksi bounding box untuk setiap objeknya.



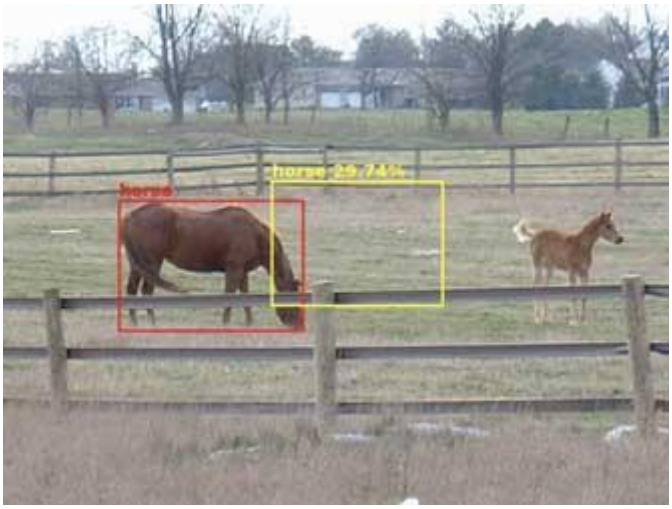
Gambar 2.3. Metode Deteksi Objek

Source: (Hollemans M, 2018)

Model deteksi objek memprediksi bounding box untuk setiap objek yang ditemukannya, serta probabilitas klasifikasi. Deteksi objek seringkali memprediksi terlalu banyak bounding boxes. Setiap kotak juga menyertakan confidence score, yang menunjukan seberapa besar kemungkinan model yakin bahwa prediksi berisi objek. Deteksi objek memiliki dua model arsitektur penyelesaian yang populer yaitu one-stage detectors dan two stage detectors(Hollemans. M, 2018).

Pada two stage detectors contohnya pada model high-end seperti Faster R-CNN. Langkah pertama yang dilakukan model ini adalah menghasilkan proposal wilayah. Proposal wilayah merupakan area gambar yang berpotensi memiliki objek dan kemudian membuat prediksi terpisah untuk masing-masing wilayah ini. Two stage detectors memiliki tingkat prediksi yang baik tetapi bekerja cukup lambat karena perlu untuk menjalankan bagian deteksi dan klasifikasi model

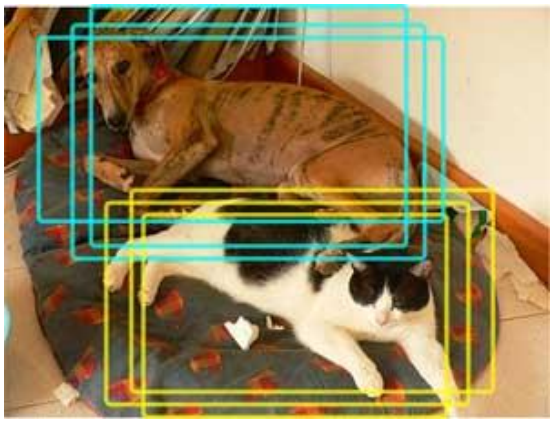
beberapa kali hingga mencapai prediksi tertinggi. Di sisi lain, one-stage detectors hanya membutuhkan satu lintasan melalui jaringan syaraf dan memprediksi semua bounding boxes dalam satu kali prediksi. One-stage detectors jauh lebih cepat dan lebih cocok untuk penggunaan mobile devices. Contoh paling umum dari one-stage detectors adalah YOLO, SSD, SqueezeDet dan DetectNet.



Gambar 2.4. Kesalahan Penglokalan Prediksi

Source: (zybuluo.com/1406657)

Dari dua model tersebut yang membuat one-stage detectors terlihat lebih unggul daripada two stage detectors. Hal itu karena one-stage detectors menggunakan fixed grid detectors. Setiap detektor mampu menangani setiap kemungkinan jenis objek di setiap lokasi yang memungkinkan dalam gambar. Model akan selalu memprediksi kotak yang berada di tengah gambar sehingga meminimalkan kesalahan jumlah pendeteksiannya. One-stage detectors seperti YOLO, SSD dan DetectNet semuanya mengatasi masalah dengan menetapkan setiap detektor bounding box ke posisi tertentu dalam gambar seperti pada ilustrasi Gambar 2.5. Dengan begitu, detektor belajar untuk mengkhususkan diri pada objek di lokasi tertentu. Dengan demikian, one-stage detectors memiliki kemampuan yang lebih baik dalam memprediksi objek bergerak seperti pada aktivitas patroli yang kondisi objek selalu bergerak sesuai kondisi lingkungannya.

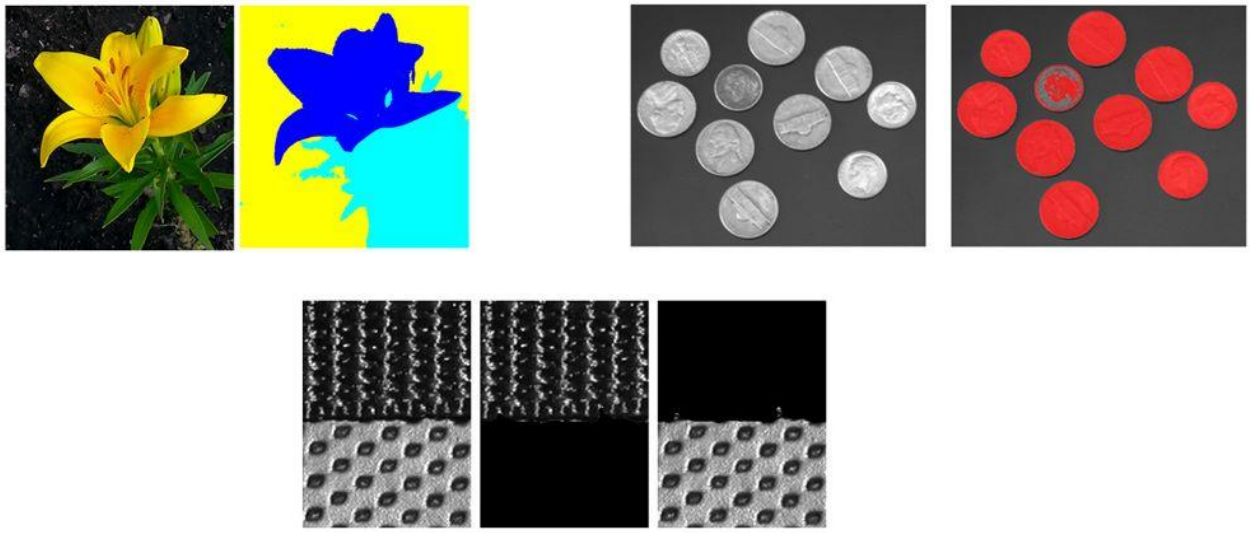


Gambar 2.5. Prediksi Overlap Source: (Hollemans M, 2018)

2.2.2. Image Segmentation

Subjek penting lainnya pada computer vision adalah segmentasi gambar. Segmentasi gambar merupakan proses membagi gambar menjadi wilayah yang berbeda berdasarkan karakteristik piksel untuk mengidentifikasi objek atau batas untuk menyederhanakan gambar dan menganalisisnya dengan lebih efisien(“Image Segmentation - MATLAB & Simulink,” 2023). Segmentasi gambar umum digunakan dalam pemrosesan dan analisis citra digital untuk mempartisi citra menjadi beberapa bagian atau wilayah, seringkali berdasarkan karakteristik piksel dalam citra atau gambar.

Proses segmentasi melibatkan pemisahan objek dari latar belakang atau pengelompokan wilayah piksel berdasarkan kesamaan warna atau bentuk. Beberapa algoritma dan teknik untuk segmentasi gambar telah dikembangkan selama bertahun-tahun menggunakan pengetahuan khusus yang secara efektif dapat menyelesaikan masalah segmentasi di area aplikasi spesifik tersebut. Aplikasi tersebut termasuk pencitraan medis, mengemudi otomatis, pengawasan video dan visi mesin. Salah satu pendekatan umum pada segmentasi gambar adalah mendeteksi kesamaan di wilayah gambar. Beberapa teknik yang mengikuti pendekatan ini adalah region growing, clustering dan thresholding yang dapat dilihat pada Gambar 2.6.



Gambar 2.6. Segmentasi Wilayah Berdasar Warna, Bentuk dan Tekstur

Source: (https://www.mathworks.com/discovery/image-segmentation.html)

2.2.3. YOLO

YOLO adalah real time object detection system dan image segmentation model(Maram et al., 2019).YOLO menggunakan algoritma computer vision yang digunakan untuk mendeteksi objek, mensegmentasi gambar, dan melokalisasi gambar pada gambar dan video secara real time. Algoritma ini dikembangkan oleh Joseph Redmon, Santosh Divvala, Ross Girshick dan Ali Farhadi yang dikembangkan pada tahun 2015-2016(Redmon et al.,2016). Keunggulan YOLO terletak pada kecepatannya dalam melakukan object detection dan image segmentation. YOLO dapat melakukan object detection pada gambar dengan kecepatan 45 frame per detik dengan GPU dan 1 frame per detik pada CPU. Selain itu, YOLO juga memiliki akurasi yang cukup baik dalam melakukan object detection pada gambar. Pada dataset COCO (Common Objects in Context), YOLO mampu mencapai mean average precision (mAP) sebesar 63.4% pada IOU (Intersection over Union) threshold 0.5(Redmon et al., 2016). YOLO menggunakan deep neural networks untuk mendeteksi objek, kemampuan YOLO dapat melakukan berbagai hal dalam mendeteksi objek seperti mendeteksi beberapa objek secara langsung, pengawasan keamanan, deteksi objek pada video hingga pengenalan wajah. YOLO memiliki keunggulan dalam hal kecepatan dan akurasi dibandingkan dengan algoritma pendeteksi objek lainnya seperti FAST R-CN, SSD dan FAST R-

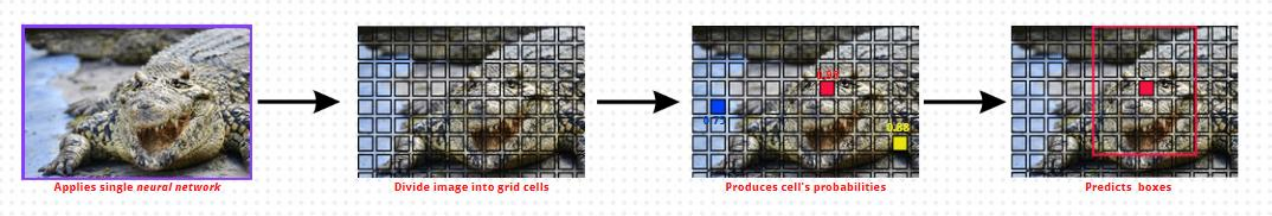
CNN menggabungkan bounding box dengan pendekatan menggunakan supresi non maksimal. Supresi non maksimal menggunakan Intersection over Union(IoU) formula untuk memprediksi probabilitas bounding box benar atau salah. Area of Intersection dan area of union didapatkan menggunakan vektor dari bounding box yang dihasilkan(Maram, Vishnoi, & Pandey, 2019).

$$\text{Intersection over Union} = \frac{\text{area of intersection}}{\text{area of union}} \quad (2.1)$$

Atau,

$$0 \leq \text{IoU} \leq 1 \quad (2.2)$$

Jika di ilustrasikan arsitektur dari cara kerja YOLO:

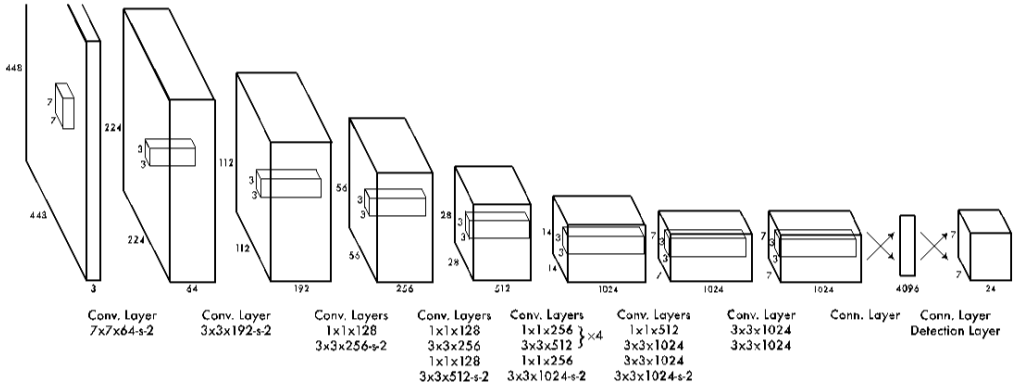


Gambar 2.7. Model Kerja Arsitektur YOLO

Source: (https://youtu.be/vRqSO6RsptU)

Berdasarkan ilustrasi pada gambar 2.7 menggambarkan bagaimana proses YOLO untuk mendeteksi gambar-gambar yang diberikan. YOLO merupakan one stage detector yang menggabungkan localization dan classification dilakukan bersamaan. Beda halnya dengan *Convlutional Neural Network*(CNN) yang menggunakan two stage detector dimana proses localization dan classification dilakukan pada jaringan yang berbeda(Wang, Bochkovskiy, & Liao, 2021). Apabila pada gambar 2.6 disegmentasikan menjadi tiga objek yang berbeda seperti air, buaya dan pasir. Maka algoritma pada YOLO akan memberitahu bahwa dalam gambar terdapat buaya atau bukan buaya dan pada gambar yang disegmentasi berbeda terdapat air dan pasir sehingga dikenal sebagai segmentasi semantik. Setelah itu YOLO akan memberitau dimana gambar buaya, pasir dan air berada yang di presentasikan di dalam grid sesuai dengan kelas yang sudah ditentukan. Oleh karena itu, maka YOLO memiliki kapabilitas untuk melakukan deteksi objek dari beberapa objek sesuai dengan kelasnya.

Semua komponen untuk deteksi objek digabungkan dalam satu jaringan syaraf. Gambar yang diciptakan dimulai dari S x S grid cells dan perlahan-lahan berkurang menjadi objek yang berada di dalam satu grid cell(Kapil Divakar, 2018). Setiap kotak pembatas diprediksi menggunakan fitur dari keseluruhan gambar. Hal ini merepresentasikan bagaimana ukuran gambar berkurang tetapi lebar dari suatu gambar bertambah dan pada akhirnya hasil dari output menciptakan probabilitas yang dibentuk oleh jaringan("GitHub - Soft-illusion/Robotics_PicoDegree," 2022). YOLO dilatih untuk mengenali 20 jenis barang atau model yang berbeda.



Gambar 2.8. Layer Konvolusi Jaringan Syaraf YOLO

Source: (Kapil Divakar, 2018)

Seperti pada Gambar 2.8. jika grid cell berisi bagian tengah objek, YOLO akan memprediksi diibaratkan bounding box 'B' untuk menyertakan objek ditengah. Selain memproduksi bounding box 'B', setiap grid cell juga akan bertanggung jawab untuk menghasilkan confidence score untuk setiap bounding box. Bounding box adalah mekanisme untuk melokalkan objek dalam gambar atau video. Bounding box hanya bisa menyertakan 1 objek dalam satu waktu.

Bounding box memiliki 4 dimensi yaitu koordinat x dan y dari pusat objek serta tinggi dan lebar kotak yang mengacu pada asal grid cell(Kapil Divakar, 2018). Setiap grid cell dari bounding box dan confidence score juga memiliki probabilitas kelas 'C' yang pada dasarnya adalah probabilitas bersyarat dari jenis objek yang ada pada bounding box yang dilambangkan sebagai **Pr(class|object).** Grid cell dapat menghasilkan probabilitas kelas 'C'. Namun, hanya terdapat 1 set

probabilitas kelas yang diprediksi per sel. Sebagai contoh satu sel hanya bisa memprediksi itu air atau pasir atau buaya dll. Grid cell tidak dapat memprediksi lebih dari 1 objek pada saat yang bersamaan. Hal ini merupakan batasan utama dari YOLO yaitu tidak dapat melokalkan dan mengidentifikasi lebih dari 1 objek per grid cell(Redmon et al.,2016).

Oleh karena itu, dimensi dari output yang dihasilkan oleh S x S grid adalah:

$$S \times S(B \times 5 + C)\text{tensor} \qquad (2.3)$$

Keterangan :

Image : S x S grids

Grid cell : 'B' = Bounding boxes and Confidence score

x, y, w, h, confidence = Pr(Object).IoU

'C' = Class probabilities w.r.t#classes

$Pr(Class_i|Object)$

Permisalan jaringan yang dievaluasi pada set PASCAL membuat kisi berukuran 7x7. Setiap grid cell memprediksi 2 bounding box dan menghasilkan 20 probabilitas kelas, output yang akan dihasilkan berupa tensor berdimensi 7x7x30(Redmon et al.,2016).

2.3. Analisis Kajian

Seiring berkembangnya urbanisasi dan digitalisasi, penelitian dan pengembangan terkait robot patroli telah beberapa kali dilakukan. Beberapa penelitian tersebut menggunakan pendalaman deep learning terutama penggunaan arsitektur computer vision yang dapat memproses satu atau sekumpulan gambar pada mesin. Berikut adalah ringkasan tabel beberapa jurnal terkait pengembangan robot patroli.

Table 2.1. Analisis Kajian Penelitian

NO	**1**	**2**	**3**
Penulis	**Sai Siddartha Maram, Tanuj Vishnoi, Sachin Pandey** (Maram et al.,	**Hou-Tsan Lee,Wei Chuan Lin, and Ching Hsiang Huang**(Lee et al., 2011)	**Yonggang Zhang , Wenqiang Li, and Feng Shen** (Zhang, Li, & Shen, 2020)

	2019)		
Judul Penelitian	Neural Network and ROS based Threat Detection and Patrolling Assistance	Indoor Surveillance Security Robot with a Self Propelled Patrolling Vehicle	Object Detection Technology of Substation Patrol Robot Based on mYOLO Algorithms
Metode Penelitian	Computer vision with YOLOv2	LAN-based with RFID and OpenCV	mYOLO, Regression Algorithm: Convolution Neural Network
Kekurangan	Akurasi pendeteksian pada kondisi kurang cahaya dan objek yang jauh masih dibawah 50%	Deteksi wajah memiliki batasan angle dan semakin besar resolusi border window akurasi akan berkurang	Tidak diterangkan
Kelebihan	Akurasi tepat saat objek berada pada jarak portrait dan dapat	Transmisi sinyal cepat karena menggunakan LAN-based system sehingga pesan langsung	Penelitian dilakukan dengan tiga jenis kamera yang berbeda dari sudut pandang yang berbeda

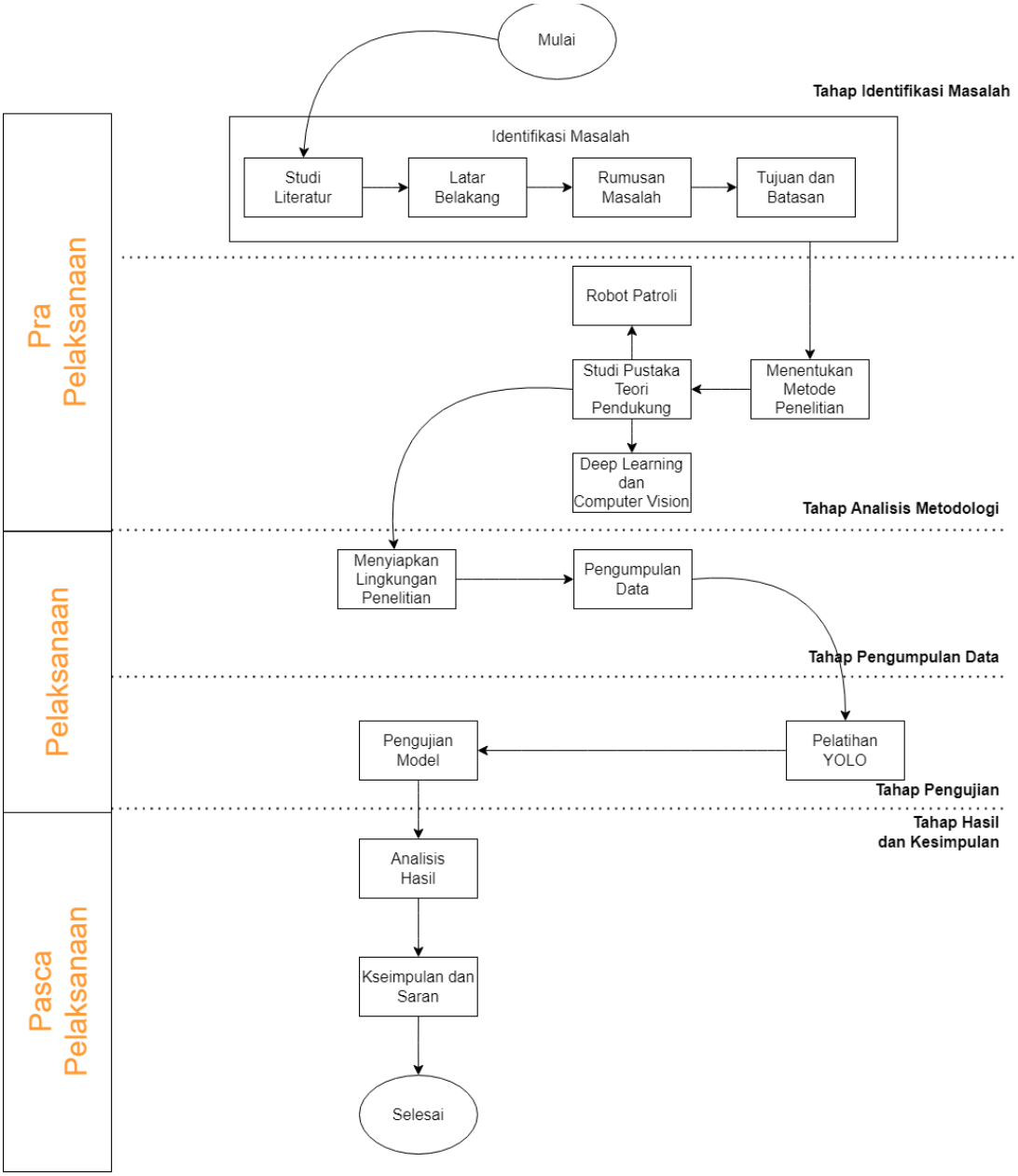
	mengenali		sehingga akurasi

	kondisi outdoor dengan baik	tersimpan dan diteruskan dari internet	identifikasi dan kecepatan deteksi sangat baik dalam segala bentuk lingkungan dan rintangan
Perbedaan	Metode penelitian menggunakan YOLOv2	Metode penelitian menggunakan sistem LAN dan OpenCV	Robot yang digunakan menggunakan metode commander and follower yang menggunakan lebih dari 1 simulasi pada robot

BAB III
METODOLOGI PENELITIAN

3.1. Tahapan Penelitian

Tahapan atau alur dari metode yang digunakan terbagi menjadi 3 alur yang telah dijelaskan pada BAB 1 terkait metodologi penelitian dan digambarkan melalui Gambar 3.1.



Gambar 3.1. Alur Metodologi Penelitian

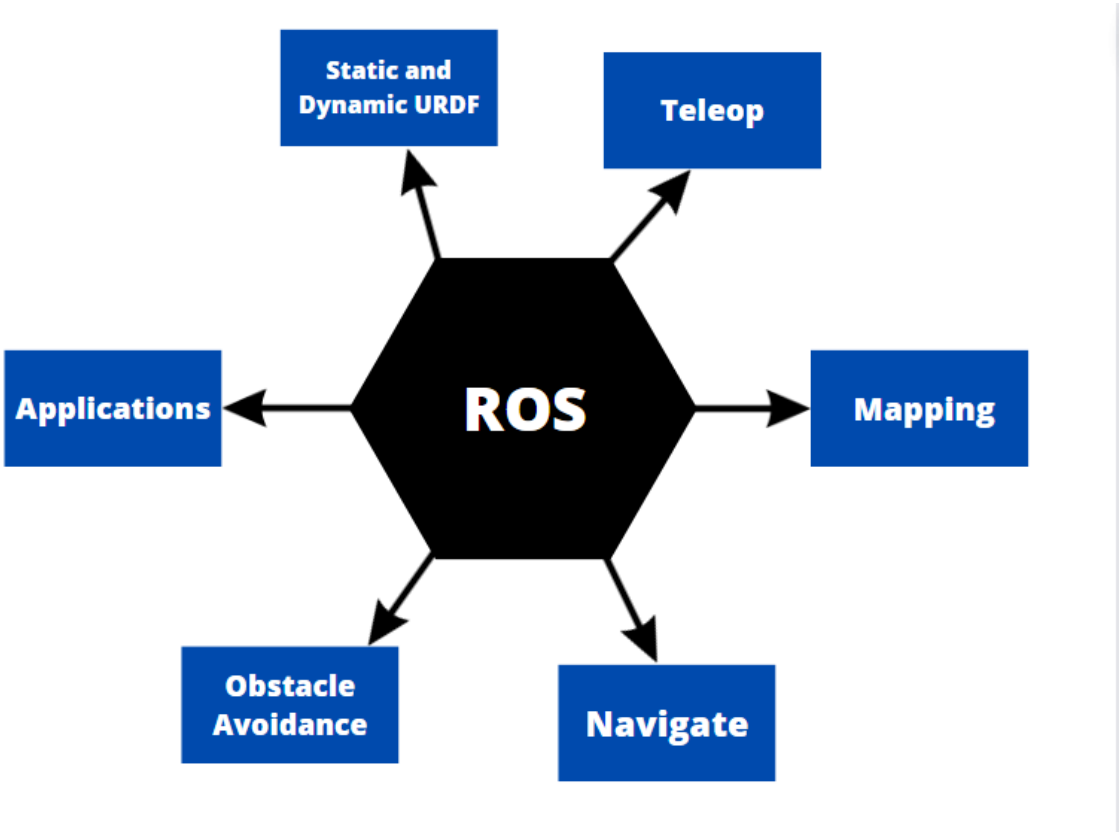
3.2. Spesifikasi Penelitian

Pada pembuatan simulasi, penulis menggunakan simulasi ”yolo_nav” yang dibangun oleh Andrii Paydin. Sebelum melakukan penelitian. Terdapat beberapa spesifikasi yang digunakan peneliti untuk menjalankan simulasi yang dicantumkan dibawah ini:

Ubuntu ≥ 20.04 LTS
Rosdistro ≥ Noetic
YOLOv5
CUDA ≥ 11.3.1
Gitpython ≥ 3.1
Matplotlib ≥ 3.3
numpy ≥ 1.18.5
opencv-python ≥ 4.1.1
Pillow ≥ 7.1.2
PyYAML ≥ 5.3.1
Requests ≥ 2.23
scipy ≥ 1.4.1
torch ≥ 1.7.0
torchvision ≥ 0.8.1
tqdm ≥ 4.64

3.3. Menyiapkan ROS

ROS merupakan pondasi dasar dalam menjalankan perangkat robot melalui komputer. Melalui terminal pada ubuntu beberapa package ROS harus diinstal untuk membuat dunia yang akan disimulasikan. Langkah-langkah yang harus disiapkan untuk membangun robot supaya dapat memiliki purwarupa seperti indra-indra yang dimiliki makhluk hidup pada umumnya yang dibangun pada langkah-langkah membuat ROS pada Gambar 3.2.



Gambar 3.2. Langkah Membangun ROS

Source: ((Robotics_PicoDegree, 2022)

Sebagai bentuk upaya bagaimana robot dapat menyamai cara berpikir manusia dan membantu manusia dalam melakukan tindakan yang repetetif seperti patroli perlu adanya manipulasi yang dibentuk dari robot agar robot memiliki kemampuan bertindak layaknya makhluk hidup. Berikut penjelasan dari ilustrasi langkah-langkah dalam membangun ROS:

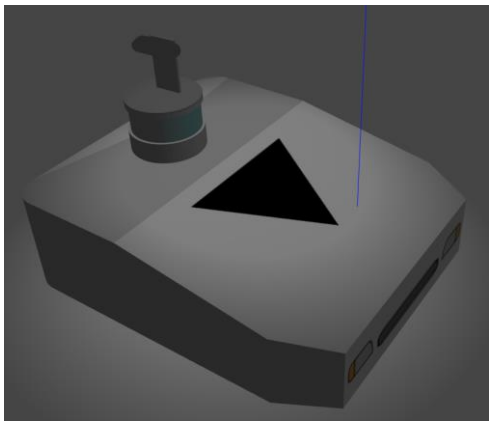
1. **Static and Dynamic URDF:** Tahapan yang pertama adalah membangun tampilan fisik dari robot, hal ini membantu ROS memahami struktur fisik robot sebenarnya dari berbagai tautan pada robot dan dimana semua sensor dan aktuator berada.
2. **Teleop:** Tahapan yang kedua adalah membangun fungsi gerak pada robot, hal ini membuat bagaimana robot dapat bergerak mengunakan sensor. Berfungsi terhadap layanan robot yang digunakan untuk mengaktifkan berbagai sensor dan memerintahkan aktuator menggunakan nilai sensor.
3. **Mapping:** Tahapan yang ketiga adalah membangun fungsi perasa pada robot, mempelajari cara agar robot dapat memahami dan merasakan lingkungannya

menggunakan LiDAR.

4. **Navigate:** Dari tahapan yang sebelumnya pada tahapan yang keempat merupakan robot yang sudah mulai memahami dunianya. Sistem memberikan perintah kepada robot menggunakan algoritma perencanaan dan pengoptimalan jalur yang berbeda untuk membuatnya berpindah ke titik tujuan, hal ini juga akan mencakup konfigurasi peta, biaya dan perencanaan yang berbeda.
5. **Obstacle Avoidance:** Tahapan kelima setelah robot melakukan navigasi dari titik A ke titik B pastinya memiliki berbagai rintangan yang telah dibentuk pada lingkungan melalui SLAM. Oleh karena itu, kita membutuhkan algoritma penghindaran rintangan yang baik dan perencanaan jalur di sekitar rintangan.
6. **Applications:** Tahapan terakhir merupakan pengaplikasian dari lima tahapan sebelumnya yang memungkinkan robot dapat melakukan tugas yang diberikan.

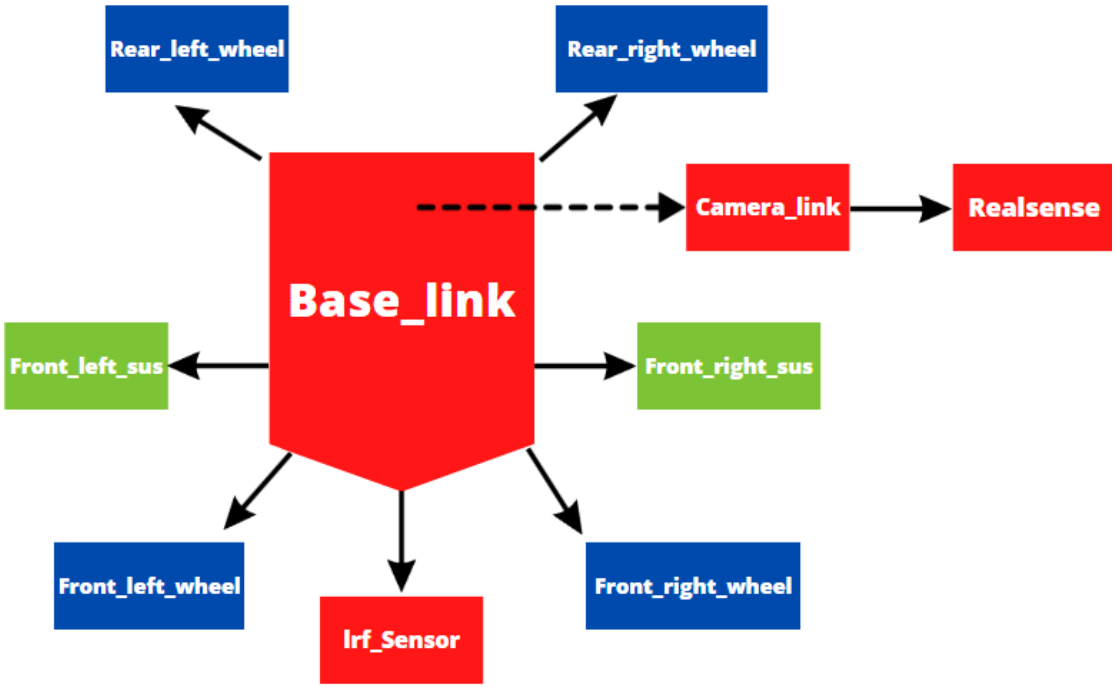
3.3.1. Static and Dynamic URDF

URDF adalah singkatan dari Unified Robot Description Format. URDF merupakan format file berbasis XML yang digunakan untuk menggambarkan properti fisik dan kinematik robot dengan cara yang dapat dibaca oleh mesin. URDF banyak digunakan dalam bidang robotika untuk berbagai aplikasi seperti simulasi robot, perencanaan gerak dan kontrol. File URDF dapat menyertakan informasi tentang tautan robot, sensor dan properti fisik lainnya seperti massa, inersia dan geometri. File URDF biasanya digunakan dengan perangkat lunak simulasi robot seperti Gazebo, ROS dan kerangka kerja pengembang robot lainnya. Pada penelitian ini peneliti menggunakan robot bernama yn_robot. Bentuk pada yn_robot ini tervisualisasikan pada Gambar 3.3.



Gambar 3.3. Robot ‘yn_robot’

Semua file URDF tersimpan menjadi satu di file XACRO. XACRO adalah XML macro language untuk membangun dan membaca file XML pada URDF. File-file URDF yang terdapat pada XACRO biasanya berisi additional features, variable names dan macros yang dapat dimuat pada Gazebo. Penjelasan terkait XACRO pada yn_robot dijelaskan secara singkat melalui illustrasi pada Gambar 3.4.



Gambar 3.4. File XACRO yn_robot

3.3.2. Teleop

Setelah membangun bentuk fisik dari robot menggunakan yn_robot. Selanjutnya adalah membangun fungsi gerak dari aktuator agar robot dapat bergerak sesuai dengan sensor pada controller yang bisa kita perintahkan dari jendela “controller” atau yang disebut dengan teleoperating.

Pada kontroler pertama terdapat ‘joint_state_controller’ yang menerbitkan gerak sendi robot pada kecepatan 50Hz.

```
joint_state_controller:
    type: joint_state_controller/JointStateController
    publish_rate: 50
```

Gambar 3.5. Konfigurasi ‘joint_state_controller’

Kemudian pada kontroler kedua adalah ‘diff_drive_controller’ yang mengendalikan pergerakan robot menggunakan differential drive system. Ada beberapa parameter didalamnya seperti mengontrol sendi roda kiri dan kanan, jarak antar roda, radius dari roda dan juga memiliki batas kecepatan dan percepatan untuk gerakan linier(x) dan sudut(z).

```
diff_drive_controller:
    type        : "diff_drive_controller/DiffDriveController"
    left_wheel  : 'left_wheel_joint'
    right_wheel : 'right_wheel_joint'
    publish_rate: 50.0
    pose_covariance_diagonal : [0.001, 0.001, 1000000.0, 1000000.0, 1000000.0,
1000.0]
    twist_covariance_diagonal: [0.001, 0.001, 1000000.0, 1000000.0, 1000000.0,
1000.0]

    wheel_separation : 0.44
    wheel_radius : 0.1

    wheel_separation_multiplier: 1.0
    wheel_radius_multiplier    : 1.0

    cmd_vel_timeout: 1.0

    base_frame_id: base_footprint

    linear:
      x:
        has_velocity_limits    : true
        max_velocity           : 0.825  # m/s
        min_velocity           : -0.825  # m/s
        has_acceleration_limits: true
        max_acceleration       : 1.0    # m/s^2
        min_acceleration       : -1.0    # m/s^2
    angular:
      z:
        has_velocity_limits    : true
        max_velocity           : 3.14   # rad/s
        min_velocity           : -3.14
        has_acceleration_limits: true
        max_acceleration       : 1.0    # rad/s^2
        min_acceleration       : -1.0
```

Gambar 3.6. Konfigurasi ‘diff_drive_controller’

Pada kontroler ketiga adalah ‘camera_stay_controller’ yang berfungsi untuk mengontrol posisi sambungan penahan pada kamera menggunakan pengontrol posisi berbasis algoritme kontrol PID(*Proportional-Integral-Derivative*).

```
camera_stay_controller:
    type    : effort_controllers/JointPositionController
    joint   : camera_stay_joint
    publish_rate: 50
    pid: {p: 1.0, i: 0.20, d: 0.0}
```

Gambar 3.7. Konfigurasi ‘camera_stay_controller’

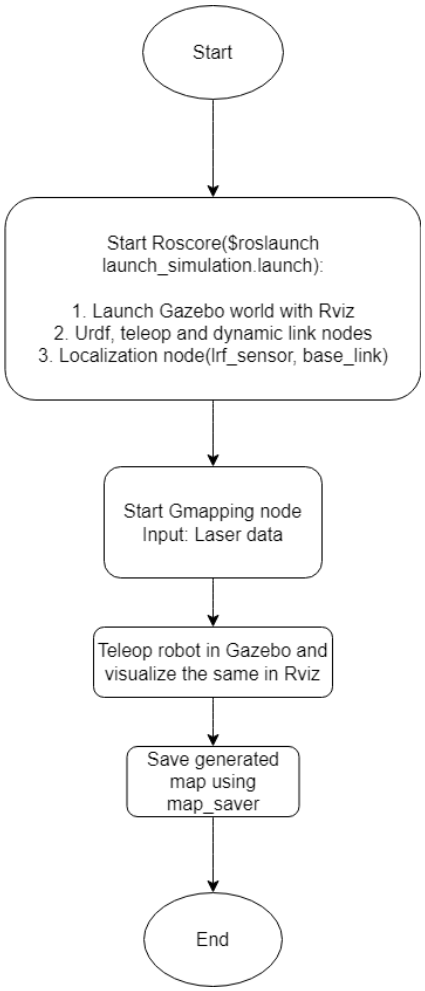
Kontroler-kontroler tersebut berfungsi untuk mengendalikan pergerakan dan posisi dari gerak sendi robot dan kamera.

3.3.3. Mapping

Gmapping adalah salah satu dari paket ROS yang menyediakan kemampuan SLAM(*Simultaneous Localization and Mapping*) untuk mobile robot. SLAM adalah proses membuat peta lingkungan yang tidak diketahui sekaligus melokalkan robot di dalam lingkungan tersebut. Gmapping menggunakan LiDAR yaitu menggunakan data dari laser range finder dan sensor odometri robot untuk membuat peta lingkungan 2D, serta memperkirakan pose robot (posisi dan orientasi) dalam lingkungan tersebut.

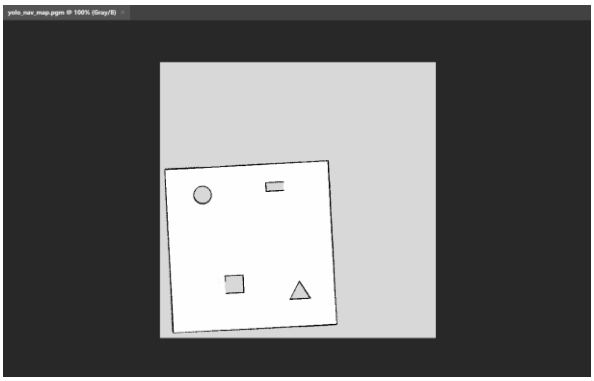
Untuk menggunakan Gmapping, sensor robot harus terkalibrasi dengan baik dan gerakan robot harus semulus mungkin. Paket tersebut membutuhkan robot untuk bergerak melalui lingkungan dengan cara mengambil sampel pada lingkungan dari berbagai sudut dan jarak. Setelah robot mengumpulkan cukup data, Gmapping dapat membuat peta lingkungan dan melokalkan robot di dalam peta tersebut.

Runtutan secara singkat proses dari pembentukan dan pembelajaran map oleh robot yn_robot dijelaskan pada diagram alur Gambar 3.8.



Gambar 3.8. Diagram Alur Proses Gmapping

Setelah seluruh map sudah ditelusuri dan dipelajari oleh robot maka file akan tersimpan oleh map_saver ke dalam format ".pgm" yang nantinya akan diimport ke dalam jendela konsol gui.

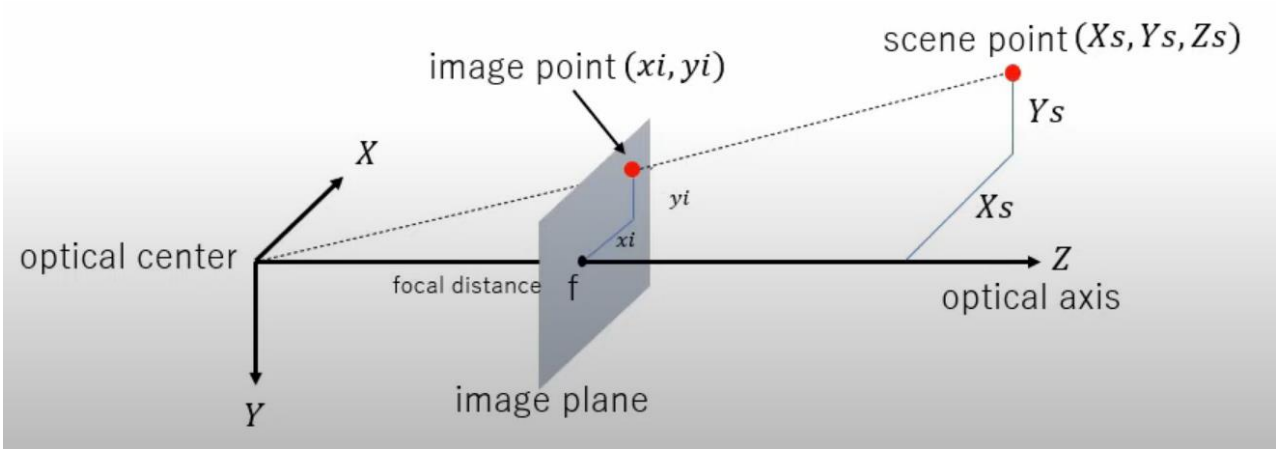


Gambar 3.9. Yolo_nav_map.pgm

3.3.4. Navigate

Setelah robot sudah memahami kondisi lingkungannya, selanjutnya merupakan bentuk perintah robot pada sistem controller agar robot dapat mengerti dan bergerak sesuai arahan konsol gui. Pada navigasi robot akan belajar bagaimana mengkalkulasi target koordinat di map menggunakan kamera yang berada pada robot. Prosedur ini terdiri dari 3 langkah.

Pada langkah pertama yaitu menghitung koordinat objek dalam koordinat sistem kamera, pada langkah ini penelitian akan menggunakan pendekatan teknik transformasi proyeksi. Untuk sepenuhnya memahami mekanisme transformasi proyeksi perspektif perlu adanya sedikit pemahaman untuk mempelajari optik. Seperti yang dicontohkan pada Gambar 3.10.



Gambar 3.10. Transformasi Proyeksi Perspektif

source: (https://youtu.be/Ob8lGOHBrig)

Model yang sangat sederhana akan digunakan untuk memahami dasar-dasar perspektif transformasi proyeksi. Dijelaskan pada Gambar 3.10 Bahwa terdapat sebuah titik di sistem koordinasi, beberapa asumsi dibuat oleh pusat proyeksi bertepatan dengan asal-usul dunia sumbu camera axis dengan sumbu z-axis.

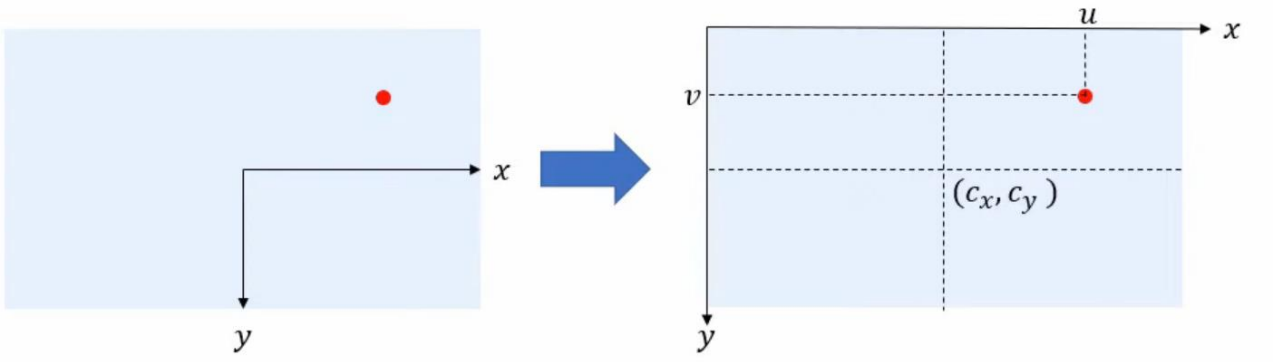
Untuk menyederhanakan turunan dari persamaan proyeksi perspektif diasumsikan bahwa bidang bayangan ada di depan dari pusat proyeksi. Tujuannya untuk mencari koordinat dari titik yang diproyeksikan pada bidang gambar menggunakan aturan persamaan segitiga x_i dan y_i.

$$\mathrm{Xi} = f\frac{\mathrm{Xs}}{\mathrm{Zs}}$$
$$\mathrm{Yi} = f\frac{\mathrm{Ys}}{\mathrm{Zs}} \qquad (3.1)$$

Keterangan:

f	: focal distance/jarak fokus
Xi, Yi	: koordinat titik gambar
Xs,Ys, Zs	: koordinat titik proyeksi

Pada langkah sebelumnya koordinat titik masuk sistem gambar kamera terhitung. Langkah selanjutnya adalah mengubahnya menjadi gambar sistem koordinat yang digunakan pada gambar pengolahan. Cara yang dilakukan adalah memindahkan asal dari pusat ke sisi kiri atas gambar yang terlihat pada Gambar 3.11.



Gambar 3.11. Transformasi Titik

source: (Wu, Hu, Hu, Li, & Lian, 2005)

Kemudian rumus turunan untuk menghitung koordinat titik yang diproyeksikan ke gambar kamera.

$$u = f\frac{Xs}{Zs} + Cx$$
$$v = f\frac{Xs}{Zs} + Cy \qquad (3.2)$$

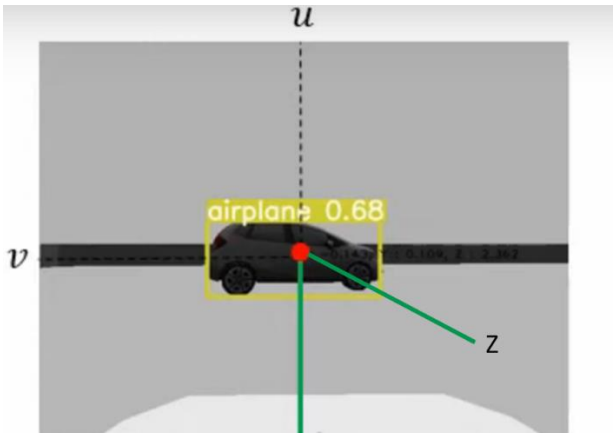
Keterangan:
f : focal distance/jarak fokus
c : optical centers
u, v : koordinat titik gambar terolah
Xs, Ys, Zs : koordinat titik proyeksi

Jika mengubah X_s dan Y_s sebelumnya maka persamaan satu akan didapat.

$$\left.\begin{aligned} Xs &= Z\frac{u - Cx}{fx} \\ Ys &= Z\frac{v - Cy}{fy} \end{aligned}\right\} \text{Eq.1} \qquad (3.3)$$

Keterangan:
f : focal distance/jarak fokus
c : optical centers
u, v : koordinat titik proyeksi gambar
Z : nilai kedalaman gambar pada titik pusat
Xs, Ys : koordinat titik tertampil

Koordinat Z dapat diperoleh dengan menggunakan kedalaman kamera dari hasil pengenalan koordinat dari titik pusat objek yang dapat dihitung terlihat pada Gambar 3.12. Kemudian dari nilai kedalaman piksel yang berada di titik pusat koordinat akan diperoleh nilai kedalaman Z yang akan diperlukan. Dengan mensubtitusi nilai Z ke persamaan koordinat satu maka koordinat X dan Y dapat diperoleh.



Gambar 3.12. Koordinat Titik Pusat

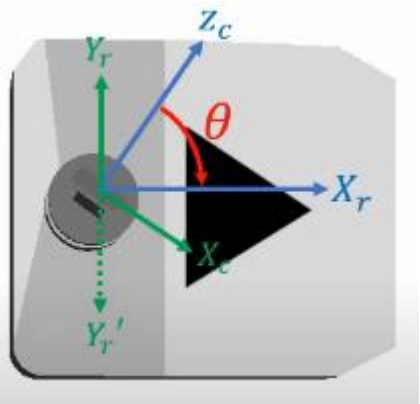
Langkah terakhir setelah mendapatkan titik-titik koordinat pada kamera. Koordinat kamera tersebut akan diubah menjadi koordinat robot, agar hal ini nantinya bisa dioperasikan menggunakan slide bar yang berada pada "controller" GUI. Setelah mendapatkan sudut rotasi theta sudut ini dapat melakukan koordinat transformasi.

$$\begin{pmatrix} Yt' \\ Xt' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} Xt \\ Zt \end{pmatrix} \quad (3.4)$$

Keterangan:

Xt' : koordinat robot terhadap X axis
Yt' : koordinat robot terhadap Y axis
Xt : koordinat kamera terhadap X axis
Zt : koordinat kamera terhadap Z axis

Dengan rumus rotasi titik pusat kamera, y-axis harus dibalik untuk dapat menyamakan koordinat robot dan koordinat kamera pada Zc axis akan menjadi koordinat robot dan koordinat kamera pada Xr axis akan menjadi koordinat robot. Kemudian, diasumsikan sudut dari kamera itu sama dengan nilai target dari GUI seperti yang dapat dilihat pada Gambar 3.13.

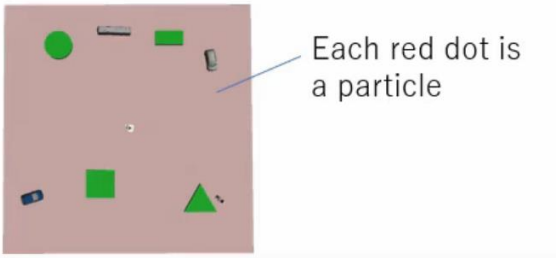


Gambar 3.13. Sudut Koordinat Kamera Robot

Setelah mengetahui koordinat robot, koordinat yang akan dihitung selanjutnya adalah koordinat suatu objek pada peta. Untuk melakukan perhitungan orientasi dan posisi robot harus diketahui terlebih dahulu. Cara untuk mendapatkan orientasi dan posisi robot menggunakan amcl yang merupakan singkatan dari Adaptive Monte Carlo Localization. Algoritma ini menyediakan paket navigasi raw yang menggunakan partikel filter untuk melacak pose robot dari lokasi yang terlacak pada map.

3.3.5. Obstacle Avoidance

Setelah robot telah memahami bagaimana cara menavigasikan dari satu titik koordinat ke titik koordinat lain, tentu saja ada beberapa halang rintang yang terdapat pada suatu lingkungan. Selanjutnya adalah bagaimana robot dapat melacak kondisi yang berada pada map. Oleh karena itu, diperlukan algoritma partikel filter. Partikel filter menggunakan bobot partikel untuk mendeteksi posisi dan orientasi robot. Hal pertama yang harus dilakukan adalah meng-inisialisasi terlebih dahulu posisi partikel yang tersebar seragam di seluruh peta seperti pada Gambar 3.14.



Gambar 3.24. Area Partikel Filter

Source: (Yuan et al., 2015)

Perhitungan posisi robot terdiri dari tiga langkah yaitu prediksi, pembaruan dan sampel ulang:

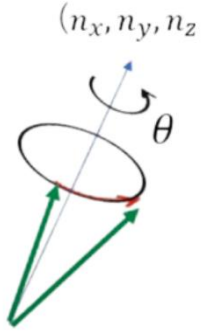
1.) Prediksi :
    Selama langkah prediksi diasumsikan robot bisa menggunakan gerakan model untuk setiap partikel, sampel dan noise pada setiap aksi model.
2.) Pembaruan :
    Pembaruan menghitung masing-masing cara partikel diperoleh. Setiap bobot partikel memiliki kesamaan untuk dapat terbaca oleh sensor dari asusmsi partikel.
3.) Sampel ulang :
    Beberapa set partikel baru dipilih dan dihasilkan melalui nilai sensor berdasarkan partikel yang tersisa dari bobot model.

Dalam kondisi dasar posisi(x,y,z) dan orientasi(x,y,z,w) robot dapat diperoleh menggunakan topik “amcl_pose”. Orientasi pada amcl dihasilkan oleh quaternion. Oleh karena itu, paternian akan dikonversi menjadi sudut euler untuk mengkalkulasi posisi target pada map. Quaternion sering digunakan untuk perhitungan yang melibatkan rotasi tiga dimensi. Quaternion umumnya diwakili pada bentuk

$$q_0 + q_1 i + q_2 j + q_3 k \qquad (3.5)$$

Keterangan:
q_0 : bilangan real kuadran I
q_1 : bilangan real kuadran II
q_2 : bilangan real kuadran III
q_3 : bilangan real kuadran IV
i, j, k : dasar quaternion

q0, q1, q2 dan q3 merupakan bilangan real sedangkan i, j dan k merupakan dasar quaternion. Setiap bilangan real merepresentasikan kuadran-kuadran yang akan digunakan untuk perkalian ke dalam persamaan bentuk dua.



Gambar 3.15. Rotasi Vektor 3D

$$\left.\begin{aligned} q_0 &= \cos\frac{\theta}{2} \\ q_1 &= n_x \sin\frac{\theta}{2} \\ q_2 &= n_y \sin\frac{\theta}{2} \\ q_3 &= n_z \sin\frac{\theta}{2} \end{aligned}\right\} \text{Eq. 2} \qquad (3.6)$$

Pada Gambar 3.15. Arah vektor pada quaternion divisualisasikan sebagai panah merah yang memutar vektor hijau sepanjang vektor arah. Kemudian untuk perkalian quaternion dapat diasumsikan bahwa terdapat dua kuadran p dan q yang akan terlihat seperti persamaan bentuk tiga.

$$p = (p_w, p_x, p_y, p_z) \text{ dan } q = (q_w, q_x, q_y, q_z) \qquad (3.7)$$

Yang didefinisikan,

$$p \times q = [p_x \quad p_y \quad p_z \quad p_w] \begin{bmatrix} q_w & -q_z & q_y & -q_x \\ q_z & q_w & -q_x & -q_y \\ -q_y & q_x & q_w & -q_z \\ q_x & q_y & q_z & q_w \end{bmatrix} \quad \text{Eq. 3} \qquad (3.8)$$

Keterangan:
p, q : bilangan real kuadran
w, x, y, z : orientasi kuadran yang dihasilkan “amcl_pose”

Setelah mendapatkan persamaan, kemudian dapat dilakukan konversi quaternion ke bentuk sudut euler. Rotasi lingkar pada x, y, z untuk α, β, γ adalah.

$$\begin{bmatrix} \cos\alpha & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$
$$= \begin{bmatrix} \cos\beta\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & \cos\alpha\sin\beta\cos\gamma - \sin\alpha\sin\gamma \\ \cos\beta\sin\gamma & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma \\ -\sin\beta & \sin\alpha\cos\beta & \cos\alpha\cos\beta \end{bmatrix} \qquad (3.9)$$

Jika rotasi vektor P menggunakan quaternion Q, maka rotasi matriks C akan menjadi seperti

$$Q \times P \times Q^{*} = (q_0 + q_1 i + q_2 j + q_3 k)(p_1 i + p_2 j + p_3 k)(q_0 - q_1 i - q_2 j - q_3 k) \quad (3.10)$$

Didapat P',

$$P' = \left[\begin{matrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_0 q_2 + q_1 q_3) \\ 2(q_0 q_3 + q_1 q_2) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_0 q_1 + q_2 q_3) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{matrix}\right] \left[\begin{matrix} p_1 \\ p_2 \\ p_3 \end{matrix}\right] \Bigg\} C \quad (3.11)$$

Keterangan:

P : sudut vektor

Q : quaternion

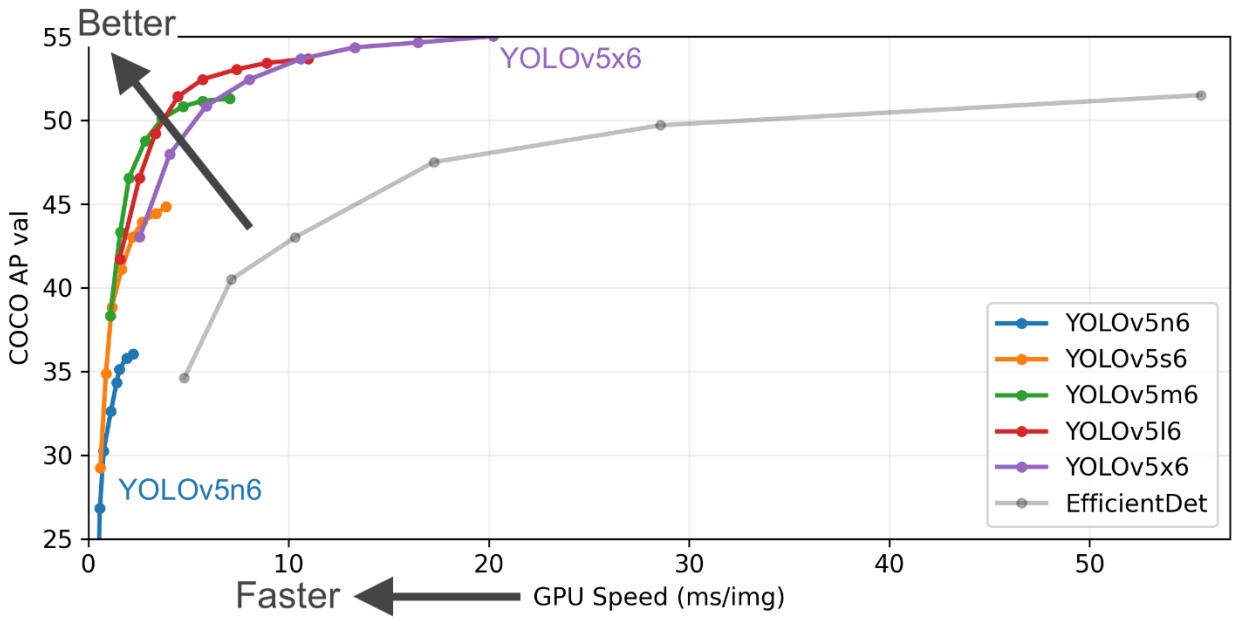
C : rotasi matriks

Setelah mendapatkan nilai rotasi matriks C maka sudut-sudut euler α, β, γ bisa didapatkan dari persamaan quaternion dan quaternion aksen seperti contoh dibawah:

$$\tan\alpha = \frac{\sin\alpha\cos\beta}{\cos\alpha\cos\beta} = \frac{2(q_0 q_1 + q_2 q_3)}{q_0^2 - q_1^2 - q_2^2 + q_3^2} \quad (3.12)$$

3.4. Pengumpulan Data

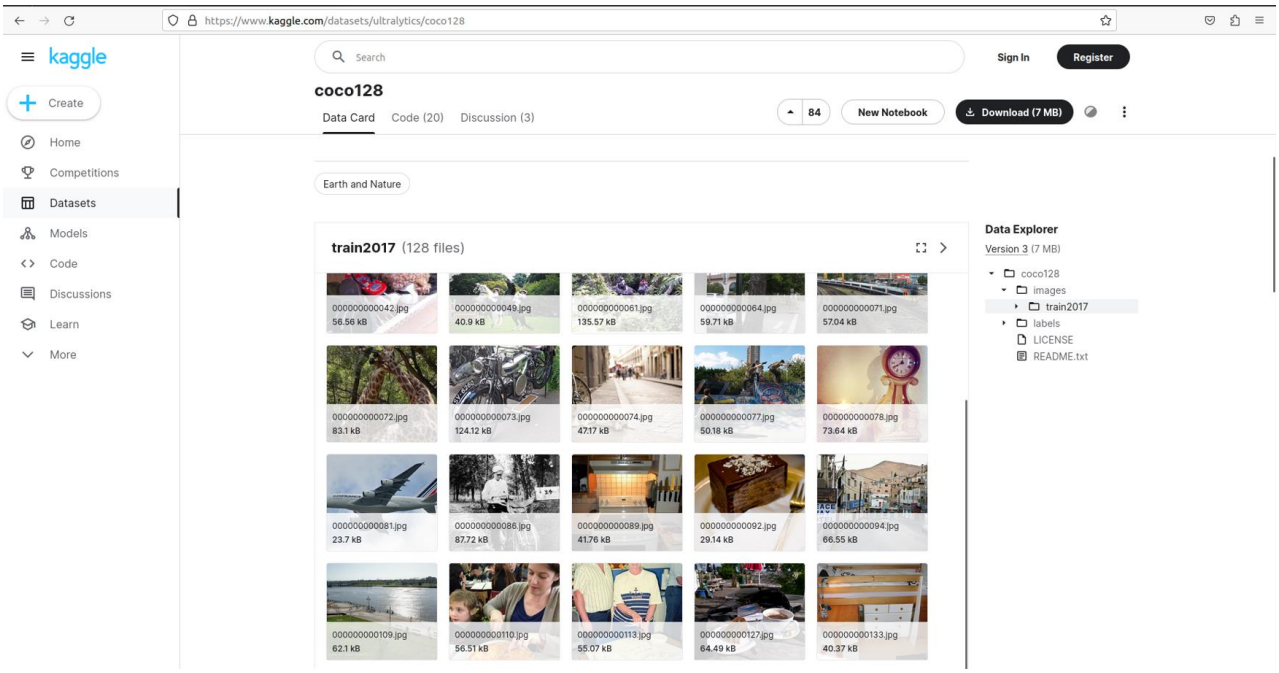
Setelah ROS beserta package didalamnya berhasil terkonfigurasi seluruhnya. Selanjutnya adalah pengumpulan data dari model. Model dataset yang digunakan pada penelitian kali ini adalah YOLOv5.



Gambar 3.16. YOLOv5 Figure

Source: (https://github.com/ultralytics/yolov5)

YOLOv5 dirancang agar mudah untuk dipelajari bagi pemula dan data yang diambil berasal dari data-data hasil dunia nyata. Adapun dataset yang digunakan merupakan dataset dari ultralytics COCO128.



Gambar 3.17. COCO128 Dataset

Source: (https://www.kaggle.com/datasets/ultralytics/coco128)

Dataset ini berisi 128 gambar pertama dari COCO2017 untuk setiap kelasnya. Dataset menggunakan 128 gambar yang sama untuk dilatih dan diuji. Hal ini dimaksudkan untuk memvalidasi alur pelatihan dengan mendemonstrasikan overfitting pada kumpulan data kecil sebelum melatih kumpulan data yang lebih besar, dataset ini berisikan 80 kelas yang terdapat pada Gambar 3.18.

```
# Classes
nc: 80  # number of classes
names: [ 'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train',
'truck', 'boat', 'traffic light',
         'fire hydrant', 'stop sign', 'parking meter', 'bench', 'bird', 'cat',
'dog', 'horse', 'sheep', 'cow',
         'elephant', 'bear', 'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag',
'tie', 'suitcase', 'frisbee',
         'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball
glove', 'skateboard', 'surfboard',
         'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon',
'bowl', 'banana', 'apple',
         'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut',
'cake', 'chair', 'couch',
         'potted plant', 'bed', 'dining table', 'toilet', 'tv', 'laptop', 'mouse',
'remote', 'keyboard', 'cell phone',
         'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock',
'vase', 'scissors', 'teddy bear',
         'hair drier', 'toothbrush' ]  # class names
```

Gambar 3.18. List Kelas COCO128

3.5. Applications

Setelah semua tahapan berhasil diciptakan, langkah terakhir adalah penerapan atau penggabungan robot patroli dengan YOLOv5 agar dapat berjalan pada simulasi. Pada pengaplikasian simulasi akan dijalankan beberapa skenario pengujian robot patroli terhadap lingkungan simulasi yang diujikan terhadap model manusia dan mobil berwarna netral dan tidak netral dalam jarak 1 – 10 meter pada sumbu Z. Adapun 3 skenario yang dijalankan diantaranya:

1. **Skenario 1** : YOLOv5 akan mendeteksi objek tunggal dengan model manusia atau mobil dengan jarak koordinat yang berbeda, pendeteksian ini membuktikan bahwa robot patroli dengan YOLOv5 mampu mendeteksi dan mengidentifikasi model tunggal dengan akurasi diatas 50% dan dengan penamaan label model yang tepat.
2. **Skenario 2** : YOLOv5 akan mendeteksi objek ganda dengan model manusia atau mobil dengan jarak koordinat yang berbeda, skenario ini akan membuktikan bahwa robot patroli dengan YOLOv5 mampu mendeteksi dan mengidentifikasi beberapa model pada satu bingkai kamera dengan akurasi diatas 50% dan dengan penamaan label model yang tepat.

3. **Skenario 3** : YOLOv5 akan mendeteksi objek manusia bergerak dengan interupsi objek disekitarnya, skenario ini akan membuktikan bahwa robot patroli dengan YOLOv5 mampu mendeteksi objek manusia berjalan dengan kondisi terhalang rintangan dan objek yang dideteksi hanya terlihat sebagian dari bentuk objeknya saja dengan akurasi daitas 50%.

BAB IV
HASIL DAN PEMBAHASAN

4.1. Batasan Implementasi

Terdapat beberapa batasan implementasi pada penelitian ini yaitu:

a) Pengujian robot patroli sebatas simulasi dan tidak menggunakan purwarupa perangkat keras

b) Kualitas kamera tidak terganggu oleh kondisi lingkungan

c) Simulasi robot patroli berfokus pada deteksi objek model manusia dan mobil

d) Koordinat ruangan sudah diketahui dan ditentukan navigasi pada GPS ruang simulasi

4.2. Pelatihan Model

Pada pelatihan model akan melewati beberapa tahapan. Tahapan pertama merupakan validasi model. Validasi akurasi model pada pemisahan ‘val’ atau ‘test’ set data COCO128. Model akan didownload secara otomatis dari rilisan versi YOLOv5 terbaru. Pada Gambar 4.1. Terdapat kode yang menjalankan validasi dari model pra-pelatihan YOLOv5 pada COCO128, validasi dataset menggunakan ‘val.py’ script.

```
!python val.py --weights yolov5s.pt --data coco.yaml --img 640 --half
```

Gambar 4.1. Kode Validasi Data

Validasi dilakukan dengan memberikan bobot pra-pelatihan untuk model YOLOv5 menggunakan argumen ‘--weights’ yang mengambil file dengan ekstensi .pt sebagai file yang bersisi bobot. Argumen ‘—data’ digunakan untuk menentukan file konfigurasi untuk dataset yang digunakan. Dalam hal ini adalah file coco.yaml. Argumen ‘--img’ digunakan untuk menentukan ukuran gambar masukan. Di sini diatur 640. Menggunakan resolusi 640 merupakan resolusi asli dari data latih COCO128. Oleh karena itu, gambar akan diubah ukurannya agar memiliki dimensi maksimal 640 piksel. Argumen ‘—half’ digunakan untuk menentukan bahwa floating-point(FP16) harus digunakan untuk inferensi yang dapat menyebabkan waktu inferensi lebih cepat pada perangkat keras yang kompatibel.

Secara keseluruhan, kode pada Gambar 4.1 Menjalankan validasi model YOLOv5 yang telah dilatih sebelumnya pada dataset validasi COCO128 untuk mengevaluasi kinerjanya. Ringkasan pada validasi dapat dilihat pada Gambar 4.2.

```
YOLOv5s summary: 213 layers, 7225885 parameters, 0 gradients
val: Scanning /content/datasets/coco/val2017... 4952 images, 48
backgrounds, 0 corrupt: 100% 5000/5000 [00:02<00:00, 2024.59it/s]
val: New cache created: /content/datasets/coco/val2017.cache
                 Class     Images  Instances          P          R
mAP50   mAP50-95: 100% 157/157 [01:25<00:00,  1.84it/s]
                   all       5000      36335      0.671      0.519
0.566      0.371
Speed: 0.1ms pre-process, 3.1ms inference, 2.3ms NMS per image at
shape (32, 3, 640, 640)
```

Gambar 4.2. Hasil Validasi

Kode ini menunjukkan hasil evaluasi model deteksi objek YOLOv5 pada dataset validasi COCO128. Evaluasi dilakukan dengan menggunakan rata-rata metrik presisi rata-rata(mAP) dengan ambang persimpangan yang berbeda di atas gabungan (IoU), ukuran objek yang berbeda (kecil, sedang, besar) dan deteksi maksimum yang berbeda per gambar (1, 10, 100) terlihat pada Gambar 4.3.

```
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] =
0.572
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] =
0.402
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] =
0.211
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] =
0.423
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] =
0.489
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] =
0.311
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] =
0.516
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] =
0.566
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] =
0.378
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] =
0.625
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] =
0.722
```

Gambar 4.3. Hasil mAP Validasi Data

Langkah selanjutnya adalah pelatihan model setelah berhasil tervalidasi. Latih model YOLOv5 pada dataset COCO128 dengan '—datacoco128.yaml' mulai dari '—weights yolov5s.pt' yang telah dilatih sebelumnya atau dari '—weights' '—cfg yolov5s.yaml'. Model pra-pelatihan akan diunduh secara otomatis dari YOLOv5 terbaru. Hasil pelatihan nantinya akan disimpan ke run/train/ dengan direktori run yang bertambah yaitu run/train/exp2, run/train/exp3 dll. Pemuat Data Mosaic digunakan untuk pelatihan yang menggabungkan 4 gambar menjadi 1 mosaik. Kode latih dapat dilihat pada Gambar 4.4.

```
# Train YOLOv5s on COCO128 for 3 epochs
!python train.py --img 640 --batch 16 --epochs 3 --data coco128.yaml --
weights yolov5s.pt --cache
```

Gambar 4.4. Kode Latih Data

Hasil dari kode pelatihan dapat dilihat pada Gambar 4.5.

```
Plotting labels to runs/train/exp/labels.jpg...
Image sizes 640 train, 640 val
Using 2 dataloader workers
Logging results to runs/train/exp
Starting training for 3 epochs...

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances
Size
        0/2      3.91G    0.04618    0.07209    0.01703        232
640: 100% 8/8 [00:09<00:00,  1.17s/it]
                 Class     Images  Instances          P          R
mAP50   mAP50-95: 100% 4/4 [00:01<00:00,  2.01it/s]
                   all        128        929      0.667      0.602
0.68       0.45

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances
Size
        1/2      4.76G    0.04622    0.06891    0.01817        201
640: 100% 8/8 [00:02<00:00,  3.78it/s]
                 Class     Images  Instances          P          R
mAP50   mAP50-95: 100% 4/4 [00:01<00:00,  2.16it/s]
                   all        128        929      0.709      0.645
0.722      0.478

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances
Size
        2/2      4.76G     0.0436     0.0647    0.01698        227
640: 100% 8/8 [00:01<00:00,  4.19it/s]
                 Class     Images  Instances          P          R
mAP50   mAP50-95: 100% 4/4 [00:01<00:00,  2.95it/s]
                   all         128         929       0.761       0.647
0.725       0.49
```

Gambar 4.5. Hasil Latih Data

Diatas merupakan hasil dari output pelatihan model deteksi objek YOLO. Berikut rincian outputnya:

1. Epoch = jumlah pelatihan epoch
2. GPU_mem = memori GPU yang digunakan selama pelatihan
3. Box_loss = kerugian untuk bounding box
4. Obj_loss = kerugian untuk objectness score
5. Cls_loss = kerugian untuk class prediction
6. Instances = jumlah instance(objek) dalam batch saat ini

7. Class, images , P, R, mAP50, dan mAP50-95 = metrik untuk mengevaluasi performa model “Class” menunjukkan kelas objek yang terdeteksi, “Images” menunjukkan jumlah gambar dalam kumpulan data, “P” menunjukkan presisi, “R” menunjukkan penarikan kembali, “mAP50” menunjukkan presisi rata-rata pada 50%, dan “mAP50-95” menunjukkan presisi rata-rata dari 50% hingga 95%. Metrik ini dihitung pada setiap pelatihan epoch.

Kemudian hasil pelatihan dari 5 kelas teratas dapat dilihat pada Gambar 4.6.

```
Model summary: 157 layers, 7225885 parameters, 0 gradients, 16.4
GFLOPs
                 Class     Images  Instances          P          R
mAP50   mAP50-95: 100% 4/4 [00:06<00:00,  1.56s/it]
                   all        128        929      0.759      0.646
0.734       0.49
                person        128        254      0.857      0.706
0.805      0.525
               bicycle        128          6      0.773      0.577
0.725      0.414
                   car        128         46      0.664      0.435
0.551       0.24
             motorcycle        128          5      0.587        0.8
0.837      0.635
              airplane        128          6          1      0.989
0.995      0.715
                   bus        128          7      0.635      0.714
0.753      0.651
                 train        128          3      0.686      0.333
0.72      0.504
                 truck        128         12      0.604      0.333
0.472      0.259
                  boat         128            6       0.938       0.333
0.449      0.177
```

Gambar 4.6. Hasil Pelatihan 5 Kelas Teratas

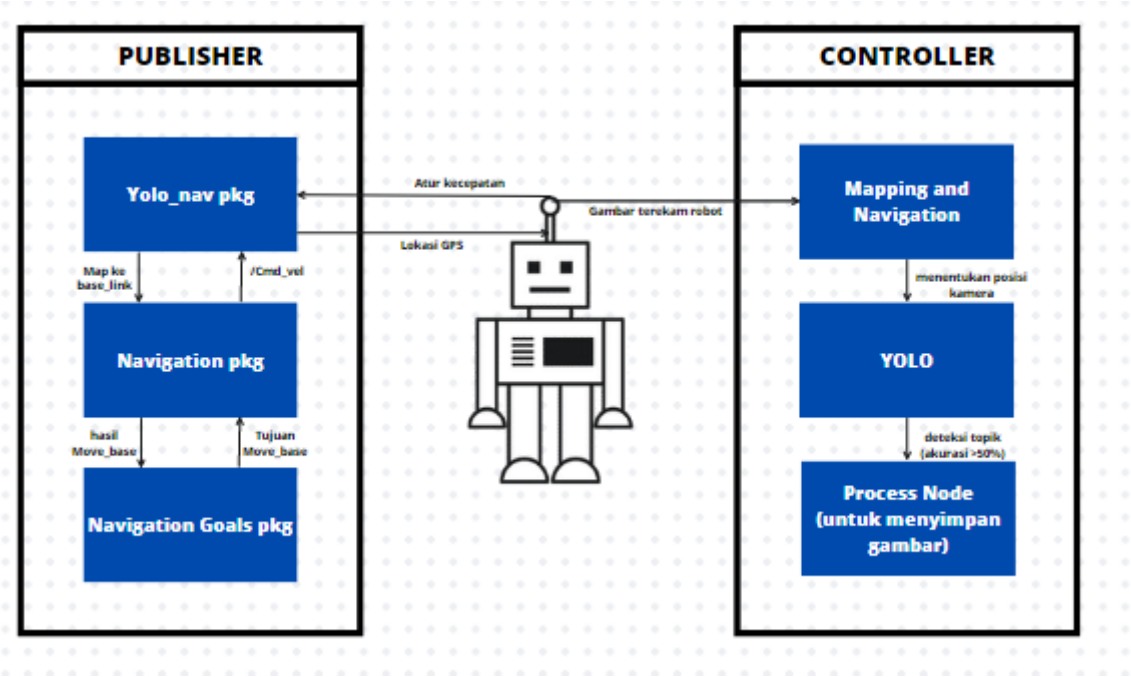
Langkah terakhir adalah pengujian pada data yang telah dilatih. Percobaan ini akan menggunakan detect.py yang tersedia pada YOLOv5 untuk menjalankan inferensi pada YOLO dan menyimpan hasilnya pada runs/detect yang dapat dilihat pada Gambar 4.7.

```
!python detect.py --weights yolov5s.pt --img 640 --conf 0.25 --
                        source data/images
# display.Image(filename='runs/detect/exp/zidane.jpg', width=600)
```

Gambar 4.7. Kode Percobaan Deteksi Objek YOLOv5

4.3. Pengujian Model

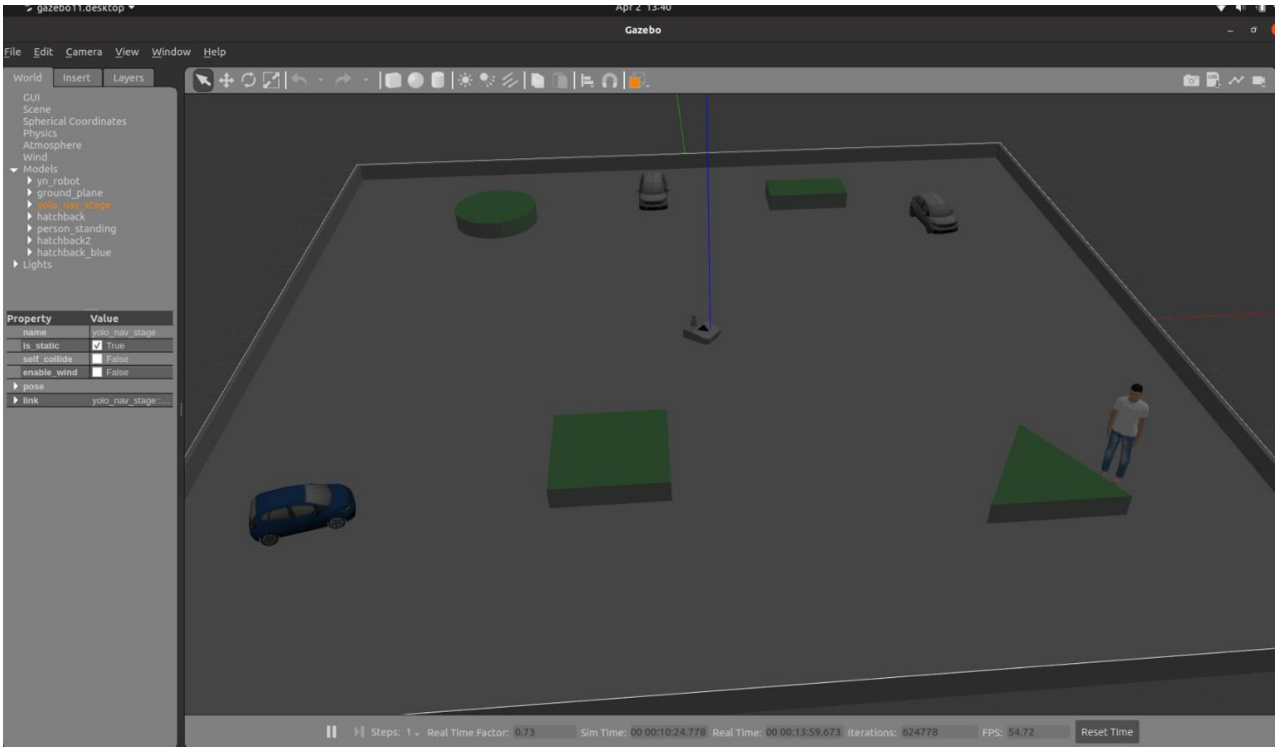
Setelah semua persiapan dan percobaan dilakukan maka langkah selanjutnya adalah pengujian YOLO terkait model yang telah diatur pada ROS. Untuk melakukan penelitian maka kita perlu mengkompilasi repositori ROS yang akan digunakan pada OS linux. Pada penelitian kali ini akan digunakan repositori yolo_nav yang memiliki robot navigasi yn_robot yang telah terancang kamera dengan konfigurasi YOLO. Rangkaian proses pengkompilasian repositori ROS terdapat pada illustrasi activity diagram deteksi objek robot dengan YOLO Gambar 4.8.



Gambar 4.8. Activity Diagram Deteksi Objek Robot dengan YOLO

Pada Gambar 4.8 dijelaskan bahwa aktivitas pada robot patroli terdapat tujuan dari navigasi yang tertampil pada lingkungan yang akan diterima oleh robot kemudian akan disampaikan oleh robot ke dalam controller gui yang nantinya dapat diatur oleh peneliti pada jendela kamera gui. Setelah objek tertampil pada kamera gui. Selanjutnya, YOLOv5 akan mendeteksi objek dan mensegmentasikan beberapa gambar di dalam jendela kamera sehingga muncul persen akurasi dan identifikasi objek pada bounding box objek. Setelah objek terdeteksi dan teridentifikasi apabila akurasi yang tertampil mencapai lebih dari 50% maka objek akan tersimpan ke dalam folder deteksi dalam format .jpg

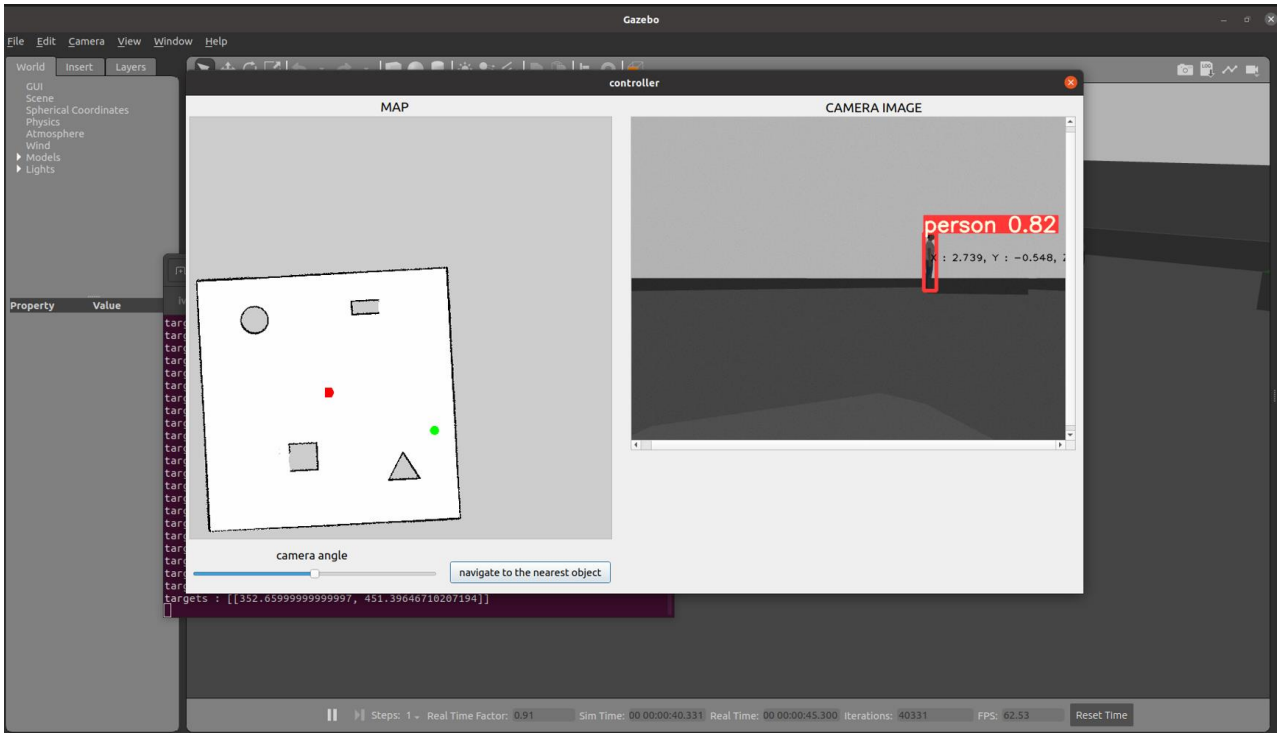
Robot patroli akan memuat lingkungannya yang telah dilakukan melalui tahapan mapping dan akan menavigasikan robot sesuai dengan arah transformasi titik koordinat.



Gambar 4.9. Tampilan Dunia Melalui Gazebo

Pada Gambar 4.9 robot memiliki lokasi dari titik koordinat semula yaitu di titik tengah dari map dengan koordinat [0,0,0] yang berjarak 5 meter keseluruh objek dengan sumbu Z dan objek yang akan deteksi berada jarak masing-masing yang telah ditentukan oleh navigasi. Dalam bagian ini akan ditampilkan koordinat target pada peta menggunakan kamera yang telah terpasang pada robot. Prosedur ini akan melibatkan tiga langkah yang telah dijelaskan pada bagian metodologi. Langkah pertama adalah menghitung objek koordinat di dalam koordinat kamera sistemnya dalam hal ini penelitian akan menggunakan perspektif Teknik Transformasi Proyeksi. Langkah kedua adalah menghitung objek koordinat dalam koordinat robot sistemnya penelitian akan menggunakan informasi dari angka sudut rotasi kamera. Langkah terakhir adalah menghitung objek koordinat dalam sistem koordinat peta sistemnya penelitian akan menemukan posisi robot dalam orientasi peta menggunakan amcl.

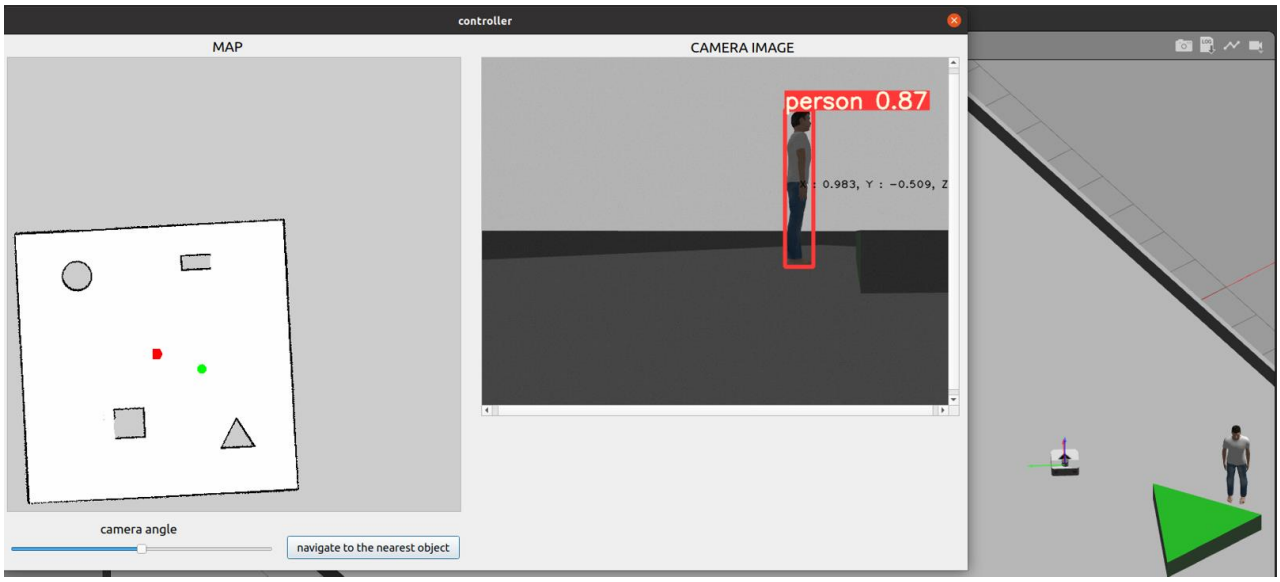
Setelah membuka tampilan dari dunia yang akan diuji, selanjutnya membuka pada terminal baru untuk memberi perintah kepada arsitektur YOLOv5 agar dapat bekerja di dalam dunia yang sudah dibentuk pada Gambar 4.10.



Gambar 4.10. Panel ‘Controller’ GUI

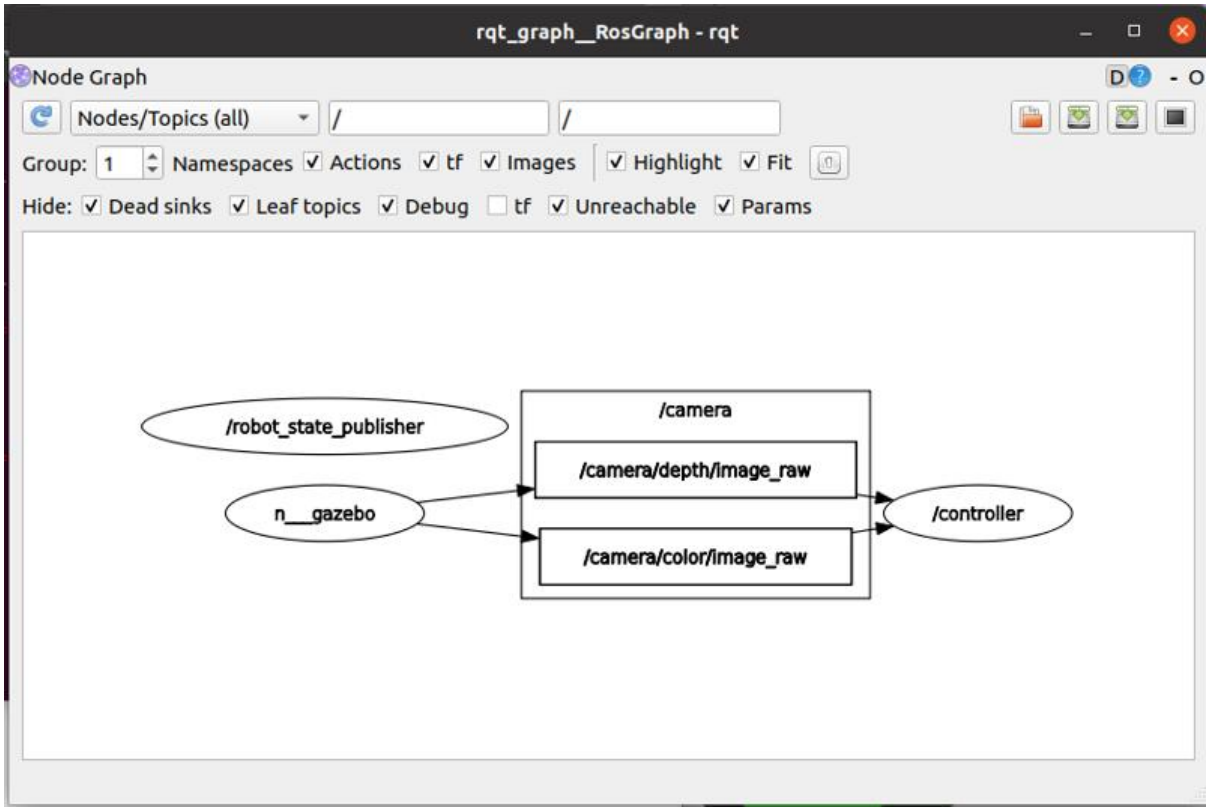
Pada Gambar 4.10. merupakan tampilan dari controller konsol GUI yang di perintah dari terminal pada kotak bagian kiri terdapat map yang sudah teridentifikasi dari proses mapping dan memuat koordinat navigasi map dan robot kemudian pada kotak bagian kanan terdapat kamera untuk deteksi objek menggunakan YOLO. Sesuai dengan yang digambarkan pada activity diagram gambar 4.8. digambarkan target tujuan pada publisher sebagai lingkaran hijau yang menandakan lokasi model terdeteksi yang dikalkulasikan menggunakan quarternion sehingga mengkonversi target posisi menjadi koordinat robot untuk menavigasikan robot sesuai dengan target yang terbit pada map.

Setelah robot patroli menerima semua koordinat yang terbaca oleh map, dengan menggunakan controller robot dapat di navigasikan ke objek terdekat yang telah terdeteksi oleh sensor sesuai Gambar 4.11.



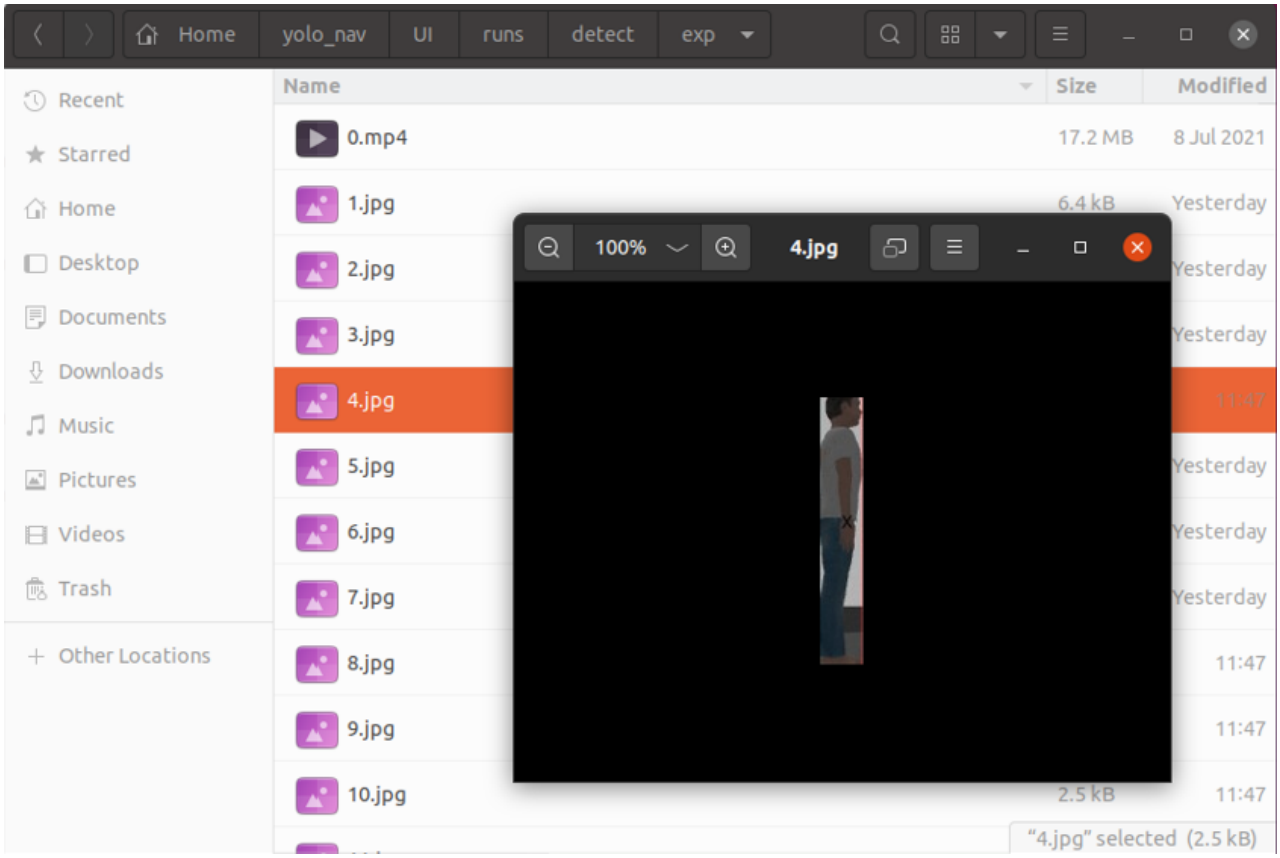
Gambar 4.11. Tampilan ‘controller’ GUI Pendeteksian Objek pada ROS

Setelah semua berjalan proses kerja robot patroli akan di debug untuk memastikan seluruh proses simulasi pada robot berjalan sesuai yang ditampilkan pada node Gambar 4.12.



Gambar 4.12. Debug Proses Penelitian

Setelah objek berhasil terdeteksi maka proses selanjutnya controller akan memproses gambar yang terdeteksi dengan YOLOv5 ke dalam pengambilan gambar yang akan di export ke dalam folder runs/detect/exp dalam bentuk format jpg seperti yang tertampil pada Gambar 4.13.



Gambar 4.13. Hasil Simpan Gambar Deteksi Objek

Pada Gambar 4.13 hasil dari deteksi objek yang tersimpan dan terdeteksi dari YOLO memiliki akurasi diatas 50% pada setiap kelasnya gambar akan dipotong dengan ukuran berskala sesuai dengan hyperparameter dari objek.

$$hyp[\text{'obj'}] \mathrel{*}= (imgsz \ / 640) ** 2 * 3. /nl \qquad (4.1)$$

Keterangan:

hyp['obj'] : Skala ke ukuran gambar dan layer

imgsz : Ukuran gambar yang terhitung dari besaran kotak

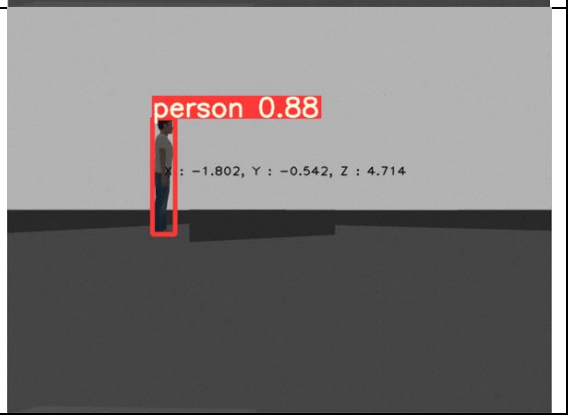
nl : Jumlah layer

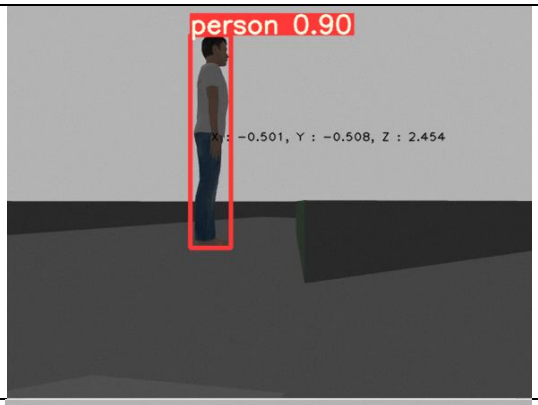
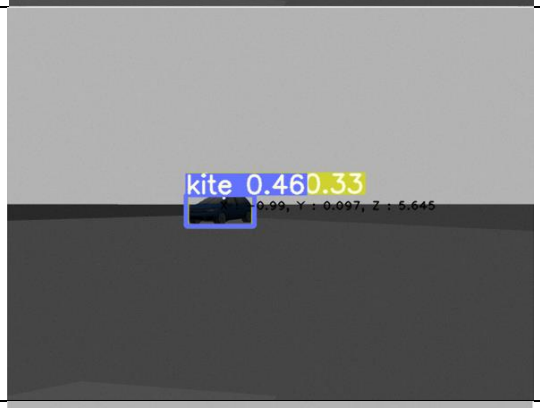
4.4. Analisis Hasil

Setelah robot patroli berhasil untuk melakukan implementasi patroli sederhana yang telah dijalankan pada simulasi, maka analisis hasil pun dapat dijalankan. Tujuan dari analisis hasil ini untuk mengukur kinerja simulasi yang telah dibuat apakah analisis dari simulasi robot patroli yang telah diteliti dapat menjawab rumusan masalah yang telah dipaparkan. Untuk itu akan dijalankan beberapa skenario yang dapat menunjukkan batasan kemampuan dari simulasi robot patroli menggunakan YOLO.

Skenario 1. YOLO akan mendeteksi objek tunggal dengan model manusia atau mobil dengan jarak koordinat yang berbeda

Tabel 4.1. Skenario 1 Pengujian

Deskripsi	Jarak (Sumbu Z)	Gambar
Model manusia berbaju putih tampak samping	Z : 6,897	
Model manusia berbaju putih tampak samping	Z : 4,714	

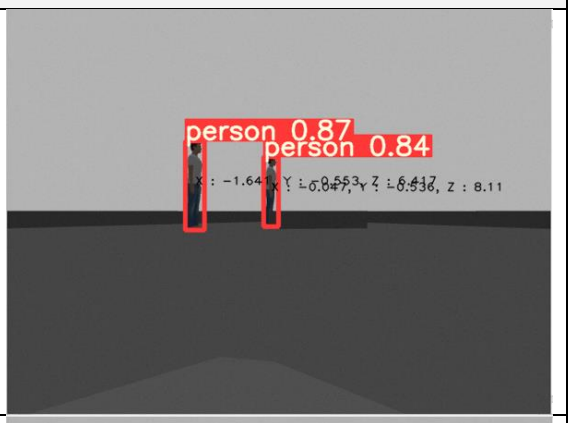
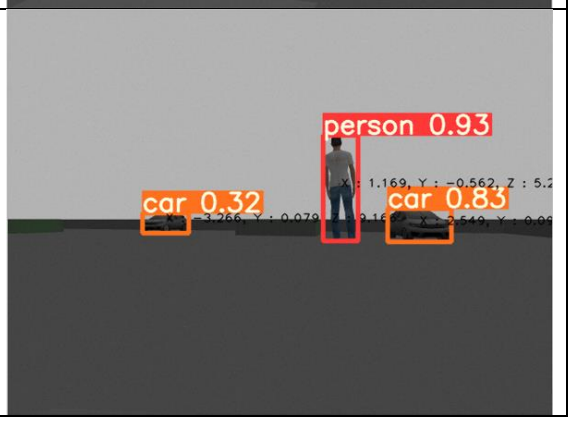
Model manusia berbaju putih tampak samping	Z : 2,454	
Model mobil berwarna biru tampak samping	Z : 5,646	
Model mobil berwarna biru tampak samping	Z : 4,089	
Model mobil berwarna biru tampak samping	Z : 3,055	

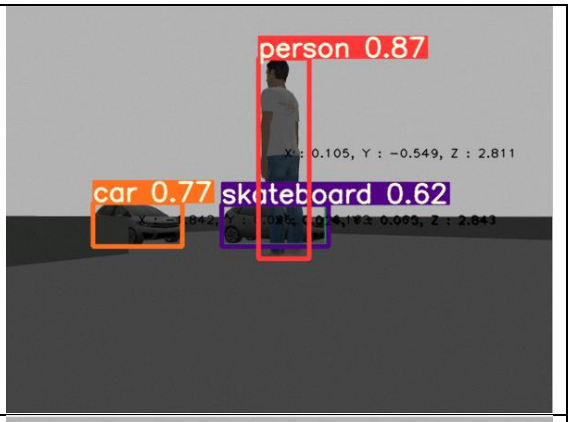
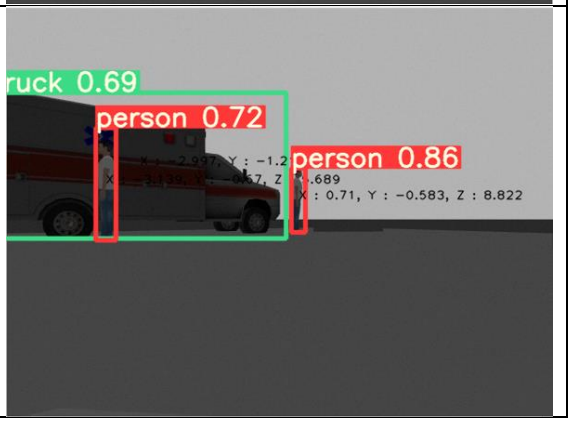
Model mobil berwarna putih tampak samping	Z : 2,581	
Model mobil berwarna putih tampak depan	Z : 5,810	

Sesuai dengan hasil pada Tabel 4.1. Pada model manusia YOLO mendapatkan akurasi pada jarak 2 meter hingga 7 meter akurasi yang dihasilkan dapat mencapai kisaran 85% hingga 90%. Kemudian, pada model mobil berwarna hasil akurasi yang dihasilkan cenderung lebih rendah dari model manusia mendapatkan kisaran 40% hingga 80%. Sedangkan, pada jarak diatas 5 meter deteksi cenderung mengidentifikasi model dengan label lain seperti pada tabel yaitu layangan. Di sisi lain untuk mobil yang berwarna putih memiliki akurasi yang lebih baik dengan akurasi diatas 50% dari berbagai sudut dan jarak.

Skenario 2. YOLO akan mendeteksi objek ganda dengan model manusia atau mobil dengan jarak koordinat yang berbeda

Tabel 4.2. Skenario 2 Pengujian

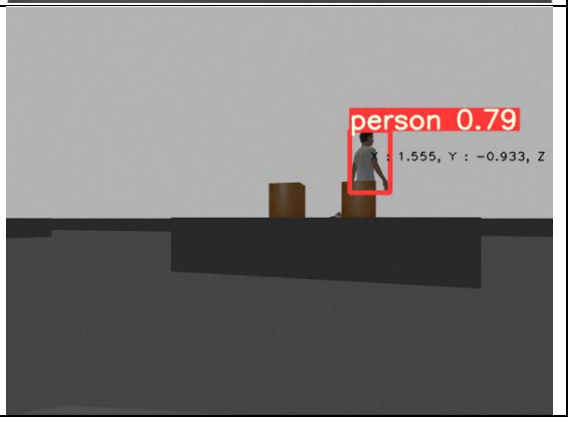
Deskripsi	Jarak (Sumbu Z)	Gambar
Model manusia berbaju putih	Z1 : 8,11 Z2 : 6,417	
Model manusia berbaju putih dan mobil berwarna biru	Z1 : 6,976 Z2 : 5,733	
Model manusia berbaju putih dan mobil berwarna putih	Z1 : 5,245 Z2 : 7,34 Z3 : 9,164	

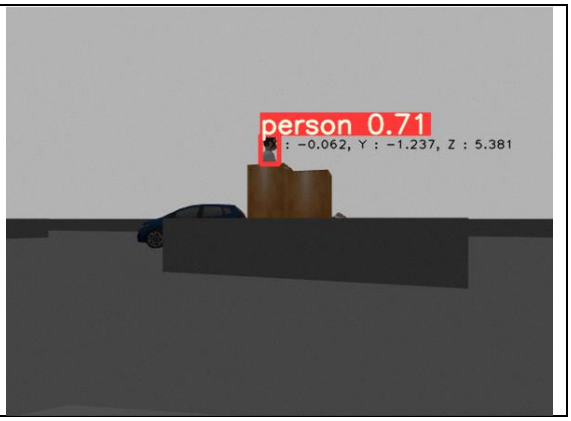
Model manusia berbaju putih, mobil berwarna biru dan mobil berwarna putih	Z1 : 2,811 Z2 : 2,643 Z3 : 4,163	
Model manusia berbaju putih dan truk ambulan berwarna putih	Z1 : 4,381 Z2 : 8,822 Z3 : 5,689	

Sesuai dengan hasil pada Tabel 4.2. YOLO berhasil mendeteksi seluruh kelas model yang berada pada jangkauan jarak tangkap kamera. Pada model manusia semua teridentifikasi dengan baik dengan akurasi rata-rata 87% pada semua jarak yang di uji cobakan. Kemudian, pada mobil berwarna biru terdapat kesalahan deteksi model yang terhalangi model manusia sehingga teridentifikasi sebagai papan seluncur. Sedangkan untuk model lainnya seperti mobil berwarna putih dan truk ambulan tidak mengalami kesalahan identifikasi dengan akurasi rata-rata diatas 50%.

Skenario 3. YOLO akan mendeteksi model manusia yang bergerak dengan interupsi dari objek-objek sekitarnya

Tabel 4.3 Skenario 3 Pengujian

Deskripsi	Jarak (Sumbu Z)	Gambar
Model manusia berjalan dan mobil berwarna biru	Z1 : 7,996 Z2 : 5,462	
Model manusia berjalan dibelakang mobil	Z1 : 8,202 Z2 : 5,465	
Model manusia berjalan tampak setengah badan dibelakang box	Z : 5,645	

Model manusia berjalan tampak kepala dibelakang box	Z : 5,381	

Sesuai dengan hasil pada Tabel 4.3. YOLO tidak memiliki masalah dalam mengidentifikasi model manusia berjalan meskipun model hanya tampak setengah badan maupun hanya terlihat kepala saja. Seluruh uji coba mengindikasikan akurasi diatas 50% pada model manusia berjalan.

Dari hasil 3 skenario yang dijalankan YOLO berhasil mendeteksi objek yang berada pada kamera dalam jarak 1 – 10 meter. Hasil skenario pendeteksian dan pelabelan pada objek manusia mendapatkan akurasi 80%-95% pada objek manusia yang tidak berpindah tempat. Sedangkan, pada percobaan objek model manusia berjalan akurasi yang didapatkan berkisar 70%-90%. Pada seluruh percobaan skenario terkait objek model manusia seluruhnya menyentuh akurasi diatas 50% sehingga seluruh data deteksi berhasil tersimpan ke dalam folder deteksi. Di samping itu, pada objek model mobil berwarna putih akurasi yang didapatkan berkisar 65%-90% dan pada mobil berwarna biru mendapatkan akurasi berkisar 40%-80%. Pada beberapa kali percobaan objek model mobil biru mendapat kesalahan pelabelan dan identifikasi dan hasil akurasi yang diberikan dapat akurat lebih dari 50% hanya pada jarak 1 – 7 meter pada objek mobil berwarna biru.

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil-hasil yang telah didapatkan, penelitian ini telah berhasil membangun dan mensimulasikan robot untuk melakukan kegiatan berpatroli sederhana yang diimplementasikan dengan computer vision dan deep learning menggunakan YOLOv5. Sehingga robot memiliki kemampuan seperti layaknya makhluk hidup yang memiliki kemampuan berupa navigasi, identifikasi dan koordinasi.

1. Penelitian ini berhasil menggabungkan robot kamera berjalan dengan model YOLOv5 untuk mendeteksi lingkungan disekitar simulasi. Dengan beberapa hasil yang telah diujikan pada hasil skenario seluruh model manusia berhasil tersimpan ke dalam folder karena memiliki tingkat akurasi diatas 50% kemudian pada model mobil tingkat akurasi masih berada dibawah 50% pada beberapa kali percobaan yang dipengaruhi oleh jarak, batas jarak yang didapatkan berada pada 1—7 meter pada objek mobil. Hal ini diharapkan dapat menjadi pendukung manusia dalam meninjau model-model yang terdeteksi pada lingkungan patroli untuk dapat ditinjau lebih lanjut serta dapat mengurangi terjadi resiko kecelakaan pada manusia saat melaksanakan patroli.
2. Penelitian ini menghasilkan deteksi yang diberikan pada model manusia mendapatkan akurasi 80% hingga 90% dalam kondisi terhalang objek atau aktivitas lain maupun dalam jarak 1 – 10 meter jauhnya. Akan tetapi, dalam pendeteksian objek lain seperti model mobil terutama pada mobil berwarna tidak netral seperti biru pelabelan YOLOv5 masih cenderung salah mengidentifikasi dan memberi label pada kelas objek pada jarak diatas 7 meter, tetapi memiliki akurasi 50% hingga 80% pada jarak 1—6 meter.

5.2. Saran

Penelitian ini menggunakan simulasi ruang sederhana dan model-model buatan. Peneliti berharap penelitian ini dapat dikembangkan oleh peneliti lain di masa mendatang dengan beberapa saran seperti:

1. Hasil deteksi objek dapat disimpan tidak hanya sekedar hasil deteksi gambar objek. Namun, bisa memuat beberapa informasi seperti waktu, tanggal dan koordinat pengambilan gambar yang menjadi satu file dengan gambar.
2. Penelitian dapat dikembangkan dan dibandingkan dengan versi arsitektur YOLO terbaru dari yang digunakan peneliti saat ini.
3. Robot patroli bisa dikembangkan lebih dari satu robot patroli dengan menggunakan metode commander and follower pada ROS. Sehingga, terdapat lebih dari satu robot yang terdistribusi dalam lingkup patroli dan robot patroli dapat berjalan sesuai pola map.

DAFTAR PUSTAKA

Alam, E., Sufian, A., Das, A. K., bhattacharya, arijit, Ali, M. F., & Rahman, M. M. H. (2022). Leveraging Deep Learning for Computer Vision: A Review. https://doi.org/10.36227/TECHRXIV.19346642.V1

Asia: Current Crime Index by City. (2022). Retrieved October 20, 2022, from https://www.numbeo.com/crime/region_rankings_current.jsp?region=142

Basilico, N. (2019). Group Robotics (M Gini and F Amigoni, Section Editors) Recent Trends in Robotic Patrolling. Current Robotics Reports, 3(2), 65-76. https://doi.org/10.1007/s43154-022-00078-5

Cao, Z., Liao, T., Song, W., Chen, Z., & Li, C. (2021). Detecting the shuttlecock for a badminton robot: A YOLO based approach. Expert Systems with Applications, 164(4), 113833. https://doi.org/10.1016/j.eswa.2020.113833

Crime in Indonesia. Safety in Indonesia. (2019). Retrieved October 20 2022, from https://www.numbeo.com/crime/in/Indonesia-Indonesia

Deep Learning in Computer Vision: Principles and Applications - Google Buku. (2022). Retrieved April 11 , 2023 from https://books.google.co.id/books?hl=id&lr=&id=10jpDwAAQBAJ&oi=fnd&pg=PP1&dq=computer+vision+%2B+robots+%2B+security&ots=wHr4IkMzV3&sig=xYTAXyedxrF8c6offy8MeUnWA8A&redir_esc=y#v=onepage&q&f=false

Divakar Kapil, Medium. (2018). Retrieved April 11, 2023, from https://medium.com/@divakar_239/yolo-v1-part-2-bfc686ae5560

GitHub - Soft-illusion/Robotics_PicoDegree. (2023). Retrieved April 11, 2023, from https://github.com/Soft-illusion/Robotics_PicoDegree

Hiatt, L. M., Harrison, A. M., & Trafton, J. G. (2011). Accommodating Human Variability in Human-Robot Teams through Theory of Mind. Twenty-Second International Joint Conference on Artificial Intelligence. Retrieved from https://www.aaai.org/ocs/index.php/IJCAI/IJCAI11/paper/view/3336

Hollemans, M. (2018). One-stage object detection. One-stage Object Detection. Retrieved April 14, 2023, from https://machinethink.net/blog/object-detection/

Image Segmentation - MATLAB & Simulink. (2023). Retrieved July 7, 2023, from https://www.mathworks.com/discovery/image-segmentation.html

Kang, H. S., Lee, D. W., & Shin, D. H. (2020). SW Development for Easy Integration of Robot System Composed of Heterogeneous Control Platforms into ROS-based System. Journal of Korea Robotics Society, 15(4), 375–384. https://doi.org/10.7746/JKROS.2020.15.4.375

Lee, H.-T., Lin, W.-C., & Huang, C.-H. (2011). Indoor Surveillance Security Robot with a Self Propelled Patrolling Vehicle. Journal of Robotics, 2011, 1–9. https://doi.org/10.1155/2011/197105

Lentin Joseph, Jonathan Cacace - Google Buku. (2018). Retrieved December 30, 2022, from https://books.google.co.id/books?hl=id&lr=&id=MulODwAAQBAJ&oi=fnd&pg=PP1&dq=robot+operating+system&ots=Cln5N5oVnP&sig=jicLvkjASxjhuZkgo86M8veFAE&redir_esc=y#v=onepage&q=robot operating system&f=false

Lopez, A., Paredes, R., Quiroz, D., Trovato, G., & Cuellar, F. (2017). Robotman: A security robot for human-robot interaction. 2017 18th International Conference on Advanced Robotics, ICAR 2017, 7–12. https://doi.org/10.1109/ICAR.2017.8023489

Maram, S. S., Vishnoi, T., & Pandey, S. (2019). Neural network and ROS based threat detection and patrolling assistance. 2019 2nd International Conference on Advanced Computational and Communication Paradigms, ICACCP 2019. https://doi.org/10.1109/ICACCP.2019.8883008

Nguyen, V. N., Jenssen, R., & Roverso, D. (2018). Automatic autonomous vision-based power line inspection: A review of current status and the potential role of deep learning. International Journal of Electrical Power and Energy Systems, 99(1), 107–120. https://doi.org/10.1016/j.ijepes.2017.12.016

Portugal, D., & Rocha, R. (2011). A survey on multi-robot patrolling algorithms. IFIP Advances in Information and Communication Technology, 349 AICT, 139–146. https://doi.org/10.1007/978-3-642-19170-1_15/COVER

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788)

Rostami, S. M. H., Sangaiah, A. K., Wang, J., & Liu, X. (2019). Obstacle avoidance of mobile robots using modified artificial potential field algorithm. Eurasip Journal on Wireless Communications and Networking, 21(1), 1–19. https://doi.org/10.1186/S13638-019-

1396-2/FIGURES/28

Santos, A., Cunha, A., & Macedo, N. (2019). Static-Time Extraction and Analysis of the ROS Computation Graph. Proceedings - 3rd IEEE International Conference on Robotic Computing, IRC 2019, 62–69. https://doi.org/10.1109/IRC.2019.00018

Suzuki, K., Machado, J., Matsuzaka, Y., & Yashiro, R. (2023). AI-Based Computer Vision Techniques and Expert Systems. AI 2023, Vol. 4, Pages 289-302, 4(1), 289–302. https://doi.org/10.3390/AI4010013

Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep Learning for Computer Vision: A Brief Review. Computational Intelligence and Neuroscience, 15(1). https://doi.org/10.1155/2018/7068349

Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2019). Scaled-YOLOv4: Scaling Cross Stage Partial Network.

Wu, Y., Hu, X., Hu, D., Li, T., & Lian, J. (2005). Strapdown inertial navigation system algorithms based on dual quaternions. IEEE Transactions on Aerospace and Electronic Systems, 41(1), 110–132. https://doi.org/10.1109/TAES.2005.1413751

Yuan, X., Yu, S., Zhang, S., Wang, G., Liu, S., Khoshelham, K., & Zlatanova, S. (2015). Quaternion-Based Unscented Kalman Filter for Accurate Indoor Heading Estimation Using Wearable Multi-Sensor System. Sensors, 15(4), 10872–10890. https://doi.org/10.3390/s150510872

Zhang, Y., Li, W., & Shen, F. (2020). Object Detection Technology of Substation Patrol Robot Based on mYOLO Algorithms. 2020 IEEE International Conference on Mechatronics and Automation, ICMA 2020, 594–598. https://doi.org/10.1109/ICMA49215.2020.9233751